NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

SIMULATION AND PERFORMANCE ANALYSIS OF THE
AD HOC ON-DEMAND DISTANCE VECTOR ROUTING
PROTOCOL FOR TACTICAL MOBILE AD HOC
NETWORKS

by

Tyrone P. Theriot

December 2000

Thesis Advisor:	Murali Tummala
Second Reader:	Robert Ives

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE December 2000	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. TITLE AND SUBTITLE : Simulation and Performance Analysis of the Ad Hoc On-Demand Distance Vector Routing Protocol for Tactical Mobile Ad Hoc Networks	5. FUNDING NUMBERS REZHB
6. AUTHOR(S) Theriot, Tyrone P.	

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) SPAWARSYSCEN, D841 (Attn: Dr. North) 53560 Hull Street, San Diego, CA 92152 - 5001	10. SPONSORING /MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	12b. DISTRIBUTION CODE

13. ABSTRACT (*maximum 200 words*)

This thesis presents a simulation and analysis of the Ad Hoc On-Demand Distance Vector Routing Protocol (AODV) for mobile ad hoc network (MANET) environments using the Network Simulator 2 (NS2) tool. AODV is being suggested for possible implementation in the Joint Tactical Radio System (JTRS) for the United States military. Utilizing an AODV model resident in NS2, the simulation focuses on key performance parameters that include the packet delivery fraction, routing loss, buffer loss, total loss, throughput and goodput. The AODV node movement and traffic connection files have been generated to measure the network performance for a given environment using specific parameters. The results reported in this thesis indicate that the network environment size, packet rate and offered load are critical to the network performance. Node velocity played a minimal role in affecting the overall network performance.

14. SUBJECT TERMS Joint Tactical Radio System, Mobile Ad Hoc Network, Network Simulator 2, Protocol Analysis, Ad Hoc On-Demand Distance Vector Routing	15. NUMBER OF PAGES
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

i

THIS PAGE INTENTIONALLY LEFT BLANK

SIMULATION AND PERFORMANCE ANALYSIS OF THE AD HOC ON-DEMAND DISTANCE VECTOR ROUTING PROTOCOL FOR TACTICAL MOBILE AD HOC NETWORKS
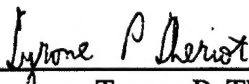
Tyrone P. Theriot
Captain, United States Marine Corps
B.S., Tulane University, 1994

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
December 2000

Author: _____
Tyrone P. Theriot

Approved by: _____
Murali Tummala, Thesis Advisor

_____
Robert Ives, Second Reader

_____
Jeffrey B. Knorr, Chairman
Department of Electrical and Computer Engineering

iii

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This thesis presents a simulation and analysis of the Ad Hoc On-Demand Distance Vector Routing Protocol (AODV) for mobile ad hoc network (MANET) environments using the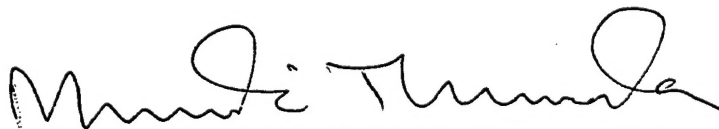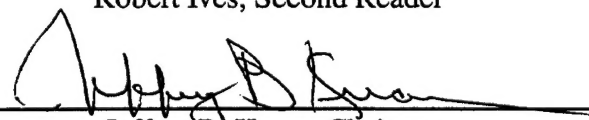 Network Simulator 2 (NS2) tool. AODV is being suggested for possible implementation in the Joint Tactical Radio System (JTRS) for the United States military. Utilizing an AODV model resident in NS2, the simulation focuses on key performance parameters that include the packet delivery fraction, routing loss, buffer loss, total loss, throughput and goodput. The AODV node movement and traffic connection files have been generated to measure the network performance for a given environment using specific parameters. The results reported in this thesis indicate that the network environment size, packet rate and offered load are critical to the network performance. Node velocity played a minimal role in affecting the overall network performance.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ABBREVIATIONS

AODV	Ad Hoc On-Demand Distance Vector Routing
ARP	Address Resolution Protocol
BPS	Bits per Second
CBR	Constant Bit Rate
COTS	Commercial-off-the-shelf
DARPA	Defense Advanced Research Projects Agency
DMR	Digital Modular Radio
DNS	Domain Name Server
DOD	Department of Defense
DSDV	Destination Sequenced Distance Vector Routing
DSR	Dynamic Source Routing Protocol
IETF	Internet Engineering Task Force
ID	Identification
IP	Internet Protocol
ISI	Information Sciences Institute
JTRS	Joint Tactical Radio System
LBNL	Lawrence Berkeley National Laboratory
MAC	Medium Access Control
MANET	Mobile Ad Hoc Network
NPS	Naval Postgraduate School
NAM	Network Animator
NS2	Network Simulator 2
OPNET	Optimum Network Performance
OTCL	Object Tool Command Language
PARC	Palo Alto Research Center
PARSEC	Parallel Simulation Environment for Complex Systems
QDR	Quadrennial Defense Review
QoS	Quality Of Service
RFC	Request For Comments
RREP	Route Reply
RREQ	Route Request
SDR	Software Defined Radio

TCL	Tool Command Language
TCP	Transmission Control Protocol
TORA	Temporally Ordered Routing Algorithm
TTL	Time to Live
UCB	University of California at Berkeley
UCLA	University of California at Los Angeles
USC	University of Southern California
VINT	Virtual InterNetwork Testbed
VTC	Video Teleconferencing
ZRP	Zone Radius Protocol

EXECUTIVE SUMMARY

The Department of Defense (DOD) is extremely interested in developing mobile wireless technology. One of the United States' most recent conflicts, Desert Storm and Desert Shield, demonstrated the necessity for effective mobile wireless voice and data communications across all services. As a result, the Joint Tactical Radio System (JTRS) Acquisition program was established to ensure that the next procurement of tactical radios would enable all services to have the ability to communicate amongst each other. The intent behind JTRS is to develop a new family of tactical radios that will be used with all U. S. armed services, will be interoperable with existing tactical radios, will be multi-band/multi-mode digital radios, will be capable of future technology insertion (wireless data networking), will be scaled for use in all environmental domains (e.g. airborne, ground, mobile, fixed station, maritime, personal communication), and will be based on a common communications system architecture. The most ambitious objective of JTRS is to exploit the radios to work effectively in a mobile ad hoc network (MANET) environment. However, conventional routing protocols are unable to meet the unique requirements of MANET. Changing topology, bandwidth and power limitations, and limited physical security combine to make the MANET a challenging environment for successful operations.

The Ad Hoc On-Demand Distance Vector Routing Protocol (AODV), developed at University of California at Santa Barbara, has been suggested for implementation in JTRS. AODV incorporates a hybrid protocol retaining specific properties from both conventional and on-demand routing protocols. From on-demand properties, AODV uses the source initiated route discovery to process destination requests outside of a one hop neighbor. From conventional properties, AODV uses sequence numbers to distinguish old routes from new ones and to guarantee loop freedom. The combination of these properties provides a stable routing protocol with an efficient method of route discovery and minimal overhead.

Utilizing a Network Simulator 2 (NS2) model of AODV, this thesis studied and examined the routing protocol's performance in three network environments. The focus of the performance analysis was on the packet delivery fraction, routing loss, buffer loss, total loss, throughput and goodput. The results provide a glimpse into the performance of AODV in three network environments chosen to represent 500m × 500m, 1500m × 1500m and 2500m × 2500m battlespaces.

The size of the network environment and the packet rate are the most critical parameters of AODV. In general, a small network environment of 500m × 500m provided the best packet delivery fraction regardless of the node velocity or the pause time. A low packet rate yielded the best results. The 1500m × 1500m and 2500m × 2500m network environments provided comparable results, but with a lower packet delivery fraction. The packet rate was the most important variable affecting the packet delivery fraction. The routing and buffer losses provided similar results. A small

network environment and low packet rate yielded the least routing and buffer loss. An increase in the packet rate or an increase in the network environment increased the routing loss. However, an increase in the packet rate alone increased the buffer loss, and an increase in the network environment alone decreased the buffer loss. The throughput remained constant depending on the packet rate and the size of the network environment. The packet rate was the most important variable affecting the throughput. The goodput revealed the most interesting results. All three network environments provided similar results. A high packet rate of either 16 or 30 packets per second yielded a goodput of relatively the same value. A low packet rate of 4 packets per second yielded a lower goodput. The offered load plot provided similar results. An increase in the offered load beyond 81 kbps yielded a similar value regardless of network environment.

The AODV model utilized in this work did not incorporate all environmental factors to adequately model the JTRS battlespace. The power consumption and physical security characteristics were not added to the simulation model. Larger network environments as well as a larger number of MANET nodes and connections are needed to accurately model the protocol behavior of AODV.

AODV is a hybrid routing protocol with potential for application in the civilian and the military environments. This thesis has provided numerous simulation results in evaluating the AODV routing protocol, and AODV is recommended as a viable routing protocol for application in a MANET environment as part of the JTRS program. Nevertheless, additional work is required to determine which MANET routing protocol has the most desirable performance for these environments.

ACKNOWLEDGMENT

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

The use of electronic services, such as electronic-mail (e-mail), file transfer protocol (ftp), video teleconferencing (VTC), etc., over a wired network has grown significantly over the last decade. These services have traditionally been implemented on wired, or static, networks to provide an effective transfer of data. The reliability of services provided to the user has increased over time to meet current demand. However, the user today requires more flexibility and requires these services in a mobile, wireless environment so as not to be limited to a fixed location. Technological advances have made mobile wireless communications possible, but with many limitations. Improvements in routing protocols are necessary for future mobile wireless communications.

The Department of Defense (DoD) is extremely interested in developing and using mobile wireless technology. One of the United States' most recent conflicts, Desert Storm/Desert Shield, demonstrated the necessity for effective mobile wireless voice and data communications across all services. As a result, the Joint Tactical Radio System (JTRS) acquisition program was established to ensure the next procurement of tactical radios would enable all services to have the ability to communicate amongst each other. The intent behind JTRS is to develop a new family of tactical radios that will be used with all U. S. armed services, will be interoperable with existing tactical radios, will be multi-band/multi-mode digital radios, will be capable of future technology insertion (wireless data networking), will be scaled for use in all environmental domains (e.g., airborne, ground, mobile, fixed station, maritime, personal communication, etc.), and will be based on a common communications system architecture. The most ambitious objective of JTRS is to exploit the radios to work effectively in a mobile ad hoc network (MANET) environment [1].

Existing routing protocols are unable to meet the requirements of the MANET environment. The MANET environment is a dynamic environment since the users move randomly throughout an area causing network topologies to constantly change, which in

1

turn degrades the performance, and constrains the bandwidth. Additionally, mobile wireless users are constrained by the battery life or power consumption for transmitting and receiving information [2]. This wide range of operating configurations poses an enormous challenge in creating an effective routing protocol for MANET environments. Existing routing protocols for the wired environment cannot overcome the randomness of the MANET environment and are extremely inefficient.

Routing protocols are generally classified as either table-driven or on-demand. Table-driven routing protocols maintain up to date routing information using internal routing tables. The overhead incurred in a wired environment is marginal. However, in a wireless environment the overhead incurred to attempt to maintain up-to-date routing information is significantly larger, constantly changes due to the random movement of the nodes and degrades the overall performance. On-demand routing protocols only create routes when desired by the source node. The source node initiates a route discovery process, and once it receives the route information it determines the best path. The route is maintained until the destination becomes inaccessible or until the route is no longer required by the source. The goal of the MANET is to incorporate the best qualities and attributes of both the table-driven and the on-demand routing protocols to create a more efficient and robust routing protocol [2].

The objective of this thesis is to perform simulations using the Ad Hoc On-Demand Distance Vector Routing Protocol (AODV) routing protocol with varied parameters and to analyze the results. The AODV routing protocol was simulated in three network environments of 500m × 500m, 1500m × 1500m and 2500m × 2500m using varied node velocities, packet rates and offered loads to determine its performance. The focus of the performance analysis was on the packet delivery fraction, routing loss, buffer loss, total losses, throughput and goodput.

This thesis is organized as follows: Chapter II begins with an introduction into the MANET environment and routing protocols. Dynamic Source Routing (DSR) and Destination Sequenced Distance Vector Routing (DSDV) are used to illustrate the difference between on-demand and table-driven MANET protocols. Chapter III

introduces the ad hoc on-demand distance vector (AODV) routing protocol and explains its method of operation. Chapter IV provides the reader with an understanding of the AODV model and describes the simulation tool "Network Simulator 2 (NS2)" used in this thesis. Chapter V presents the results of the simulation and analysis. Conclusions and recommendations are included in Chapter VI.

THIS PAGE INTENTIONALLY LEFT BLANK

II. MOBILE AD HOC NETWORK PROTOCOLS

A mobile ad hoc network (MANET) is comprised of mobile platforms (work stations, routers, etc.) that are able to move freely in a random manner. The user nodes and the infrastructure itself are constantly in a state of transition. A MANET is considered to be an autonomous system of mobile nodes. It may operate in isolation or may interface with a fixed network via a gateway. The creators of MANET envisioned the interface with a fixed network to function as a stub network which provides access to larger fixed networks. Figure 1 provides an illustration of the differences between a MANET, a Mobile IP network and a fixed network. The fixed network is stationary with
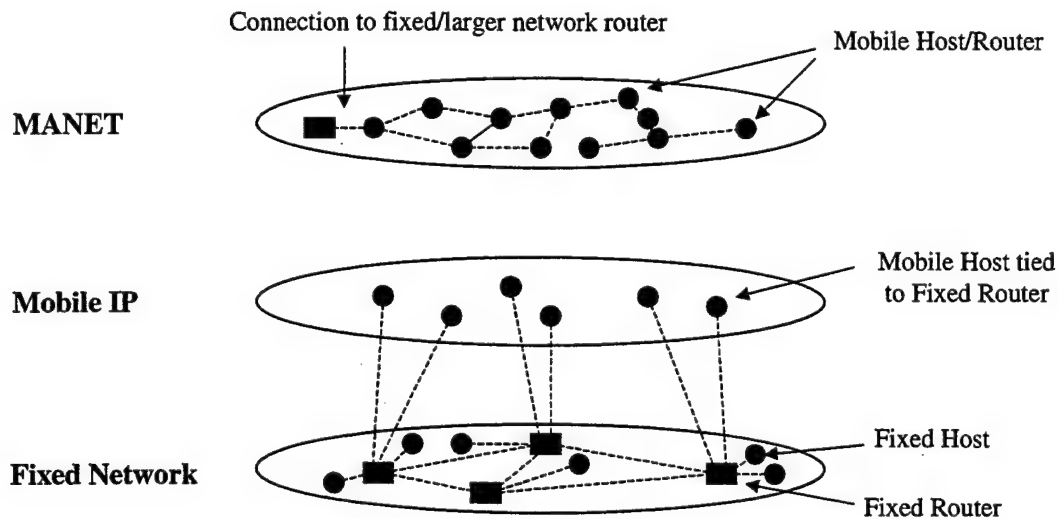
Figure 1. MANET Layer In Perspective (After Ref. [2]).

minimal host/router mobility. A Mobile IP network allows the hosts' mobility, but requires the mobile host to be tied to a fixed router in a fixed network. MANET allows the mobile host/router to move randomly thereby providing complete mobility.

A MANET has four distinct performance characteristics: dynamic topology, bandwidth constraints, energy constrained operations, and limited physical security. These performance characteristics lead to specific design criteria that a protocol must

5

adhere to in order to extend the high data rate, stability, and quality of service (QoS) capabilities of a fixed network to a MANET. Routing protocols for this unique environment must be able to conform to the performance characteristics outlined in order to be a viable solution for MANET. Existing routing protocols of the fixed network are unable to meet the performance characteistics for a MANET environment due to considerable overhead for routing tables and maintenance, and are slow to react to topological changes [2].

A.   CONVENTIONAL ROUTING PROTOCOLS

Conventional routing protocols use either link state or distance vector routing algorithms. Distance vector routing is based on the classical Bellman-Ford routing mechanism [3]. Each router contains routing information in its internal routing table. The routing table contains the distance to all available routers, where the distance is measured in hops. The table is formed by computing the shortest path to each host from the information received as well as obtaining other routers' broadcasting messages. The messages are sent by request, periodically, and with each there is a change of metric in the routing table. Link state routing is based on Dijkstra's algorithm [3]. Network topology information is used to make routing decisions. Each router actively tests the status of its link to each of its neighbors and sends this information to its other neighbors, which then propagate it throughout the autonomous system. Each router takes this information and builds a complete routing table.

Both algorithms allow a host to find the next hop neighbor to reach the destination via the shortest path. In determining the shortest path, cost measures, such as link utilization or queueing delay, are used to assist in making the determination. For ad hoc networks, these conventional shortest path routing protocols take too long to converge, have a high message complexity and do not react well to the rapidily changing topology and limited bandwidth, which requires message complexity to be kept to a minimum. The unique environment of ad hoc networks requires a new series of MANET protocols to handle this rapidly changing environment [3, 4].

6

B.   TABLE DRIVEN VERSUS ON-DEMAND PROTOCOLS

A new series of MANET protocols have emerged to challenge the existing fixed network protocols: table driven and on-demand protocols. In a table-driven routing protocol, each router maintains path information for each known destination in the network and updates its routing table entries as needed. In an on-demand routing protocol, routers maintain routing information for only those destinations that they need to contact as a source or an intermediate node for relaying of information. The basic approach consists of allowing a router that does not know how to reach a destination to send a flood search message to obtain the path information it requires. Figure 2 provides an illustration on the behavior of table-driven and on-demand protocols and the amount
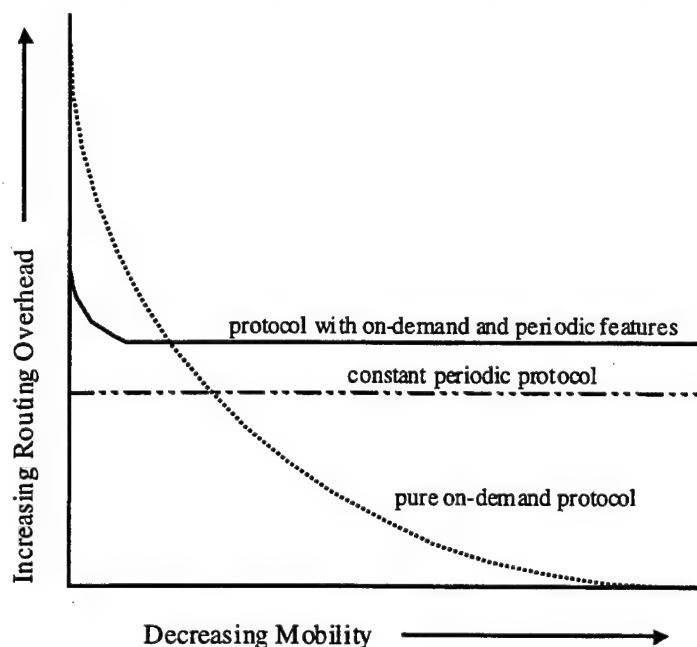
Figure 2. Behavior of On-demand and Periodic Mechanisms (From Ref. [5]).

of routing overhead associated with each one. There are numerous examples of both table-driven and on-demand routing protocols. Destination Sequenced Distance Vector (DSDV) and Dynamic Source Routing (DSR) algorithms will be explained to illustrate examples of table-driven and on-demand routing protocols. The Ad Hoc On-Demand

7

Distance Vector (AODV) routing protocol is a hybrid of DSR and DSDV and will be discussed in detail in Chapter III [3, 4].

1. Destination Sequenced Distance Vector (DSDV) Routing

Bhagwat and Perkins developed the Destination-Sequenced Distance-Vector (DSDV) routing algorithm in 1994 [6]. DSDV was based on the idea of the classical Bellman-Ford routing algorithm, which is a table-driven protocol with certain improvements. Every mobile station maintains a routing table that lists all available destinations, the number of hops to reach the destination and the sequence number assigned by the destination node. The sequence number is used to distinguish old routes from new ones and guarantees loop freedom. Routing loops cause the algorithm to continually follow an invalid route. The use of sequence numbers prevents the formation of routing loops (loop freedom). The stations periodically transmit their routing tables to their immediate neighbors. A station also transmits its routing table if a significant change has occurred in its routing table from the last update sent. Therefore, the update is both time-driven and event-driven. The routing table updates can be sent as either a full dump or an incremental update. A full dump sends the full routing table to the neighbors and could include numerous packets. An incremental update provides only those entries from the routing tables that have a metric change since the last update. Since the incremental update carries new information from the last full dump, additional space is available depending upon the metric changes. If space is available in the incremental update packet, then the new entries from the incremental update may be included with the sequence numbers that have changed. When the network is relatively stable, incremental updates are sent to avoid extra traffic, and full dumps are infrequent. In a rapidly changing network, incremental packets can grow quickly and full dumps become necessary. Each route update packet, in addition to the routing table information, also contains a unique sequence number assigned by the transmitter. The route labeled with the highest or most recent sequence number is used. If two routes have the same sequence number then the route with the best metric is used. Figure 3 illustrates the

8

routing procedure in DSDV. In this example, a packet is being sent from Node 1 to Node 3 (not shown). From Node 1, the next hop for the packet is Node 4. When Node 4 receives the packet, it looks up the destination address (Node 3) in its routing table. Node 4 then transmits the packet to the next hop as specified in the table, in this case Node 5.

A) Node 1 transmits a packet to node 4 for forwarding

B) Node 4 looks up the destination in its routing table

C) Node 4 retransmits the packet to the next hop
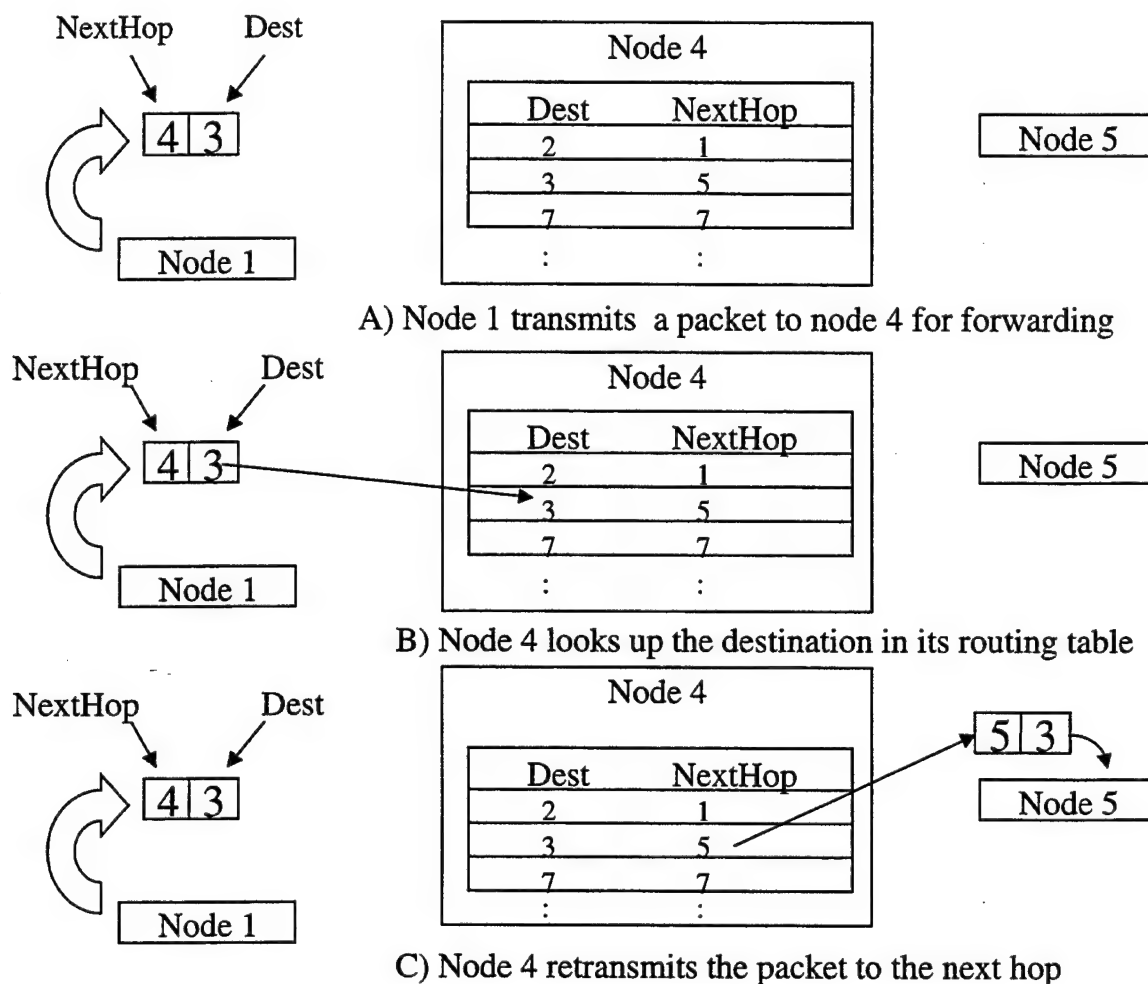
Figure 3. An Example of the Routing Procedure in DSDV (After Ref. [7]).

This procedure is repeated as required until the packet finally reaches its destination. The complexity of DSDV is in maintaining and generating routing tables. As the number of nodes in the network increases, the size of the routing tables and the bandwidth requirements increase creating significant overhead, thus increasing the time of convergence [5, 7, 9,10 and 11].

9

2. Dynamic Source Routing (DSR)

Broch, Johnson and Maltz developed Dynamic Source Routing (DSR) in 1998 [12]. DSR is based on the source routing concept: the source specifies the complete path to the destination in the packet header, and each node along this path simply forwards the packet to the next hop indicated in the path. DSR utilizes a route cache in which the source caches the routes it has acquired. A source first checks its route cache to determine the route to the destination. If a route is found, the source uses this route. If a route is not found, the source uses a route discovery protocol to discover a route. In route discovery, the source floods a query packet throughout the ad hoc network. Either the destination or another host that can complete the query from its route cache returns a reply. Each query packet has a unique identifier (ID). When receiving a query packet, if a node has already seen this ID (a duplicate ID), or it finds its own address already recorded in the list, it discards the copy and stops flooding. Otherwise, it appends its own address on the list and broadcasts the query to its neighbors. If a node can complete the query from its route cache, it may send a reply packet to the source without propagating the query packet further. Any node participating in route discovery can learn routes from passing data packets and gather this routing information into its route. Figure 4 provides an example of a packet moving through an ad hoc network with source routing. In DSR, no periodic control messages are used for route maintenance, and there is little or no routing overhead when a single or few sources communicate with infrequently accessed destinations. The disadvantage with DSR is that as the network becomes larger, control packets and message packets also become larger. Due to the limitation in bandwidth, the overhead becomes significant because of the need to carry addresses for every node in the path [5, 7, 9, 10, 11, 12 and 13].
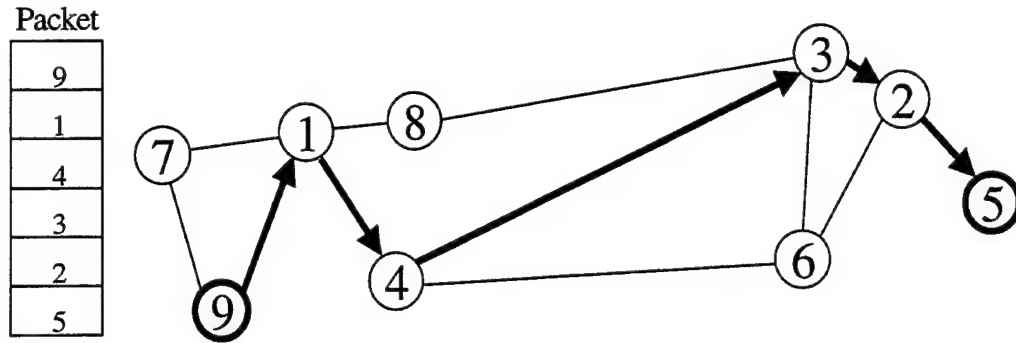
Figure 4. A Packet being Source Routed from Node 9 to Node 5 (After Ref. [7]).

C.   EVALUATION OF MANET PROTOCOLS

At the time of this writing, no specific standards exist for the evaluation of MANET protocols. Corson and Macker provide criteria for the evaluation of any MANET protocol in their RFC (RFC 2501) that was later adopted by the Internet Engineering Task Force (IETF) MANET Working Group as a defacto standard [2]. These criteria are broken down into three major categories within the MANET performance issues.   The first category consists of seven fundamental qualitative properties: distributed operations, loop freedom, demand-based operation, proactive operations, security, sleep period operation and unidirectional link support. The second category consists of four quantitative metrics to assess performance: end-to-end data throughput and delay, route acquisition time, percentage out-of-order delivery and efficiency.   The third category is made up of eight varying networking parameters: network size, network connectivity, topological rate of change, link capacity, fraction of unidirectional links, traffic patterns, mobility and fraction and frequency of sleeping modes.

Numerous issues remain to be resolved in the criteria for the evaluation of any MANET protocol.   Additionally, the number of nodes, specific use and applications required by the user will eventually determine the most appropriate protocol for these purposes. The defacto standard of the IETF for MANET protocol evaluation is still not complete. An important parameter that is not incorporated in the IETF standard is the goodput quantitative metric. The goodput metric is the measure of total packets received

11

by the destination. This metric is probably the most important since the actual amount of packets received by the destination can be evaluated to determine a true measure of routing performance. For the purpose of this thesis, AODV will be evaluated using quantitative metrics, including goodput, with varying networking parameters. The specific metrics and parameters will be discussed in detail in Chapter V.

III. AD HOC ON-DEMAND DISTANCE VECTOR ROUTING (AODV)

Perkins and Royer developed the AODV routing protocol in 1999 [14, 15]. AODV incorporates the attributes of two other routing protocols, DSR and DSDV. DSR is a source oriented on-demand protocol, and DSDV is a table-driven protocol. AODV is a combination of these two protocols and is considered to be a hybrid protocol. Specifically, AODV uses the same features as DSR for route discovery and maintenance and, from DSDV, uses the hop-by-hop routing, sequence numbers, periodic update packets and ensures loop free routing [5, 7]. The intent of this hybrid protocol is to allow local regions to utilize local routing information and to obtain a route outside the local region on-demand [14]. All routing protocols function at the network layer and the specific region is depicted in Figure 5.

Application	→	Traffic generation, QoS
Transport	→	TCP/UDP
Network	→	MANET Routing protocols, QoS
Data Link	→	MAC, QoS
Physical	→	Wireless channel, QoS
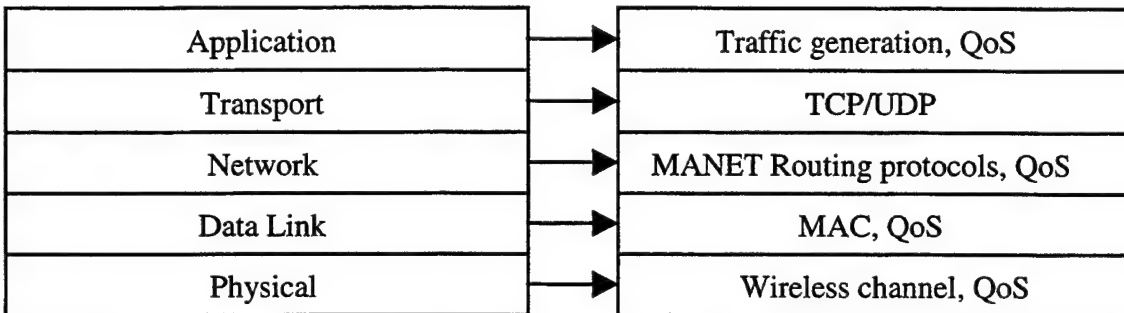
Figure 5. Protocol Stack for a MANET Node.

The following discussion on AODV will focus on four major areas: route discovery, route table management, path maintenance and local connectivity management.

A.   ROUTE DISCOVERY

The route discovery process is initiated when a source node needs to send information to a node either outside the local region or when no local routing information exists for the destination node. The source node begins route discovery by broadcasting a

13

route request (RREQ) packet to all of its neighbors. Figure 6 illustrates a source node sending a broadcast RREQ to destination node D. The RREQ packet contains the
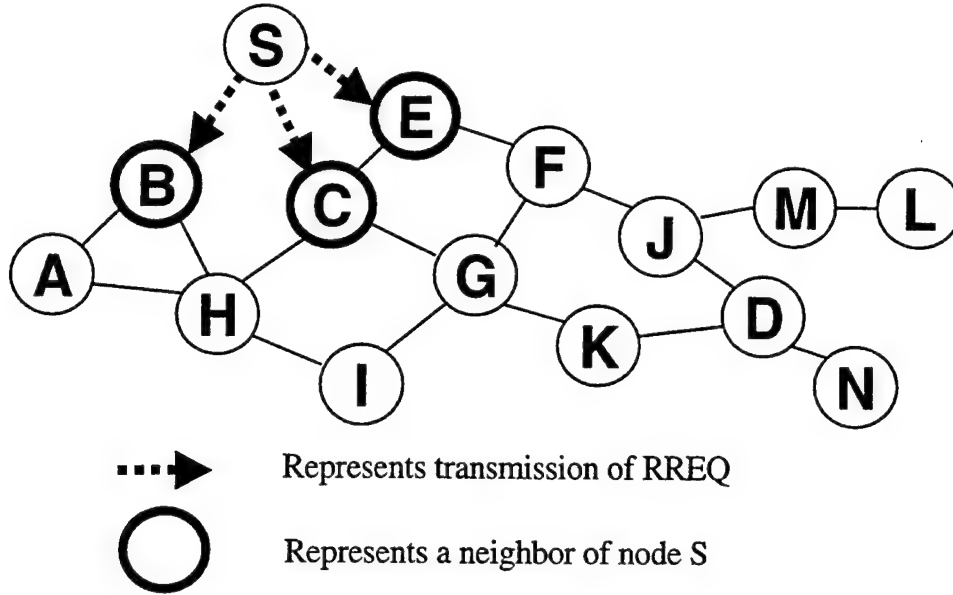
Figure 6. Initial Route Request from Node S to Node D.

following information fields: *type, reserved, hop count, broadcast ID, destination IP address, destination sequ· nce number, source IP address* and *source sequence number.* Figure 7 depicts the format of the RREQ. The *type* field indicates the type of traffic: constant bit rate (CBR) or transmission control protocol (TCP). The *reserved* field is reserved for future use. It is currently sent as 0 and ignored on reception. The *hop count* field indicates the number of hops from the source IP address to the node handling the request. The *broadcast ID* field indicates a unique sequence number identifying the particular request when taken in conjunction with the source node's IP address. The *destination IP address* field indicates the IP address of the destination for which a route is required. The *destination sequence number* field indicates the last sequence number that has been received by the source for any route towards the destination. The *source IP address* field indicates the IP address of the node that originated the request. The *source sequence number* field indicates the current sequence number for route information generated by the source of the route request.

0		1		2		3

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

Type [8]	Reserved [16]	Hop count [8]
Broadcast ID [32]		
Destination IP Address [32]		
Destination Sequence Number [32]		
Source IP Address [32]		
Source Sequence Number [32]		

Figure 7. Route Request (RREQ) Format (From Ref. [14]).

An intermediate node can reply to the RREQ if it knows an up-to-date route to the destination or it is the destination itself by sending a *route reply* (RREP) back to the source node. Figure 8 shows the format of the RREP. The *type* field indicates the type of traffic, CBR or TCP. The *L* field indicates if the L-bit is set. If set, the message is a hello message and contains a list of the node's neighbors. If not set, then the L-field is ignored. The *reserved* field is reserved for future use. It is currently sent as 0 and ignored on reception. The *hop count* field indicates the number of hops from the source IP address to the destination IP address. The *destination IP address* field indicates the IP address of the destination for which a route is supplied. The *destination sequence number* field indicates the destination sequence number associated with the route. The *lifetime* field indicates the time for which nodes receiving the reply consider the route to be valid. Otherwise, the intermediate node will rebroadcast the RREQ to its neighbors and increase the hop count. The intermediate nodes keep track of the source address and the broadcast ID and discards redundant RREQ broadcasts. If an intermediate node cannot accommodate the RREQ, it maintains the following information: *destination IP address, source IP address, broadcast ID, expiration time for reverse path route entry* and the *source node's sequence number*. This information will be necessary to implement the *reverse* and *forward path setup* that accompanies the RREP [8, 14, 15, 16 and 17].

15

0	1	2	3
0 1 2 3 4 5 6 7 8 9	0 1 2 3 4 5 6 7 8 9	0 1 2 3 4 5 6 7 8 9	0 1

Type [8]	L	Reserved [16], (includes L bit)	Hop count [8]
Destination IP Address [32]			
Destination Sequence Number [32]			
Lifetime [32]			
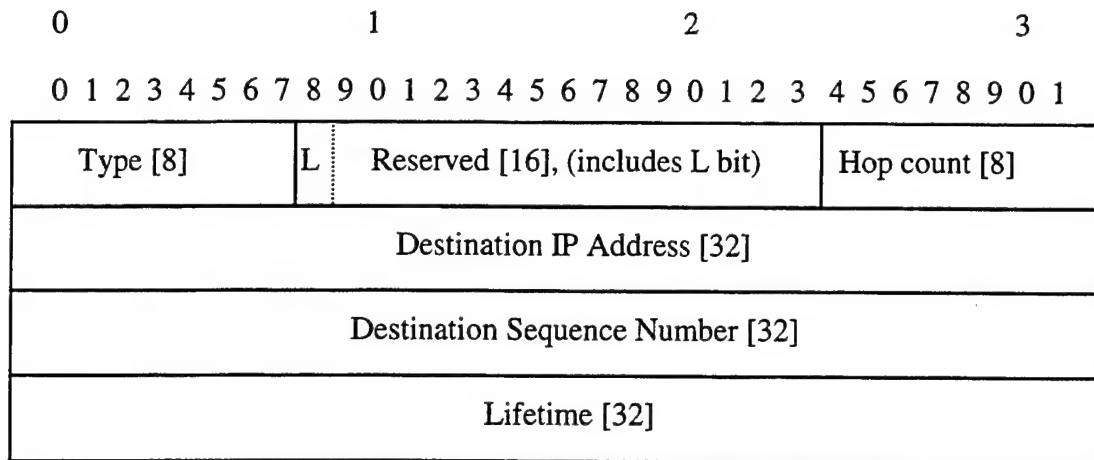
Figure 8. Route Reply (RREP) Format (From Ref. [14]).

1. Reverse Path Setup

The RREQ contains six key pieces of information. The only information of interest to the *reverse path setup* is the *source sequence number* and the *destination sequence number*. The *source sequence number* maintains the most up-to-date information concerning the reverse route to the source node. The *destination sequence number* indicates how up-to-date the route to the destination node must be in order for the source node to accept it. Figure 9 depicts the *reverse path setup* back to the source node S. The RREQ moves from the source node through numerous intermediate nodes. As the RREQ traverses the network, it automatically begins to establish the reverse path
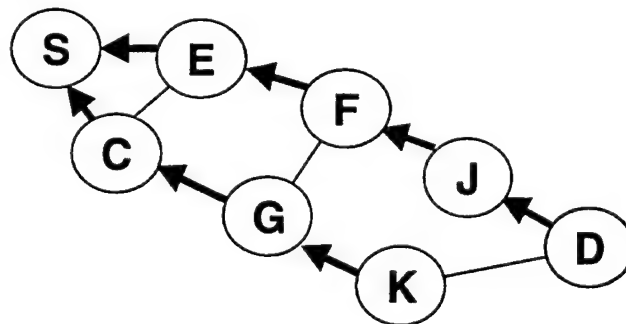
Figure 9. Reverse Path Setup to Source Node S.

from all nodes in the network back to the source node. Nodes update their route table with source node information before forwarding the RREQ. The reverse path entry is used to forward the RREP back to the source node if one is received. The expiration time of the reverse path entries is long enough for a RREP to be received and forwarded and is not a fixed value. Figure 10 is a portion of the C++ pseudocode used for *reverse path setup* resident within the Network Simulator 2 (NS2). The pseudocode describes the process of creating and ensuring the *reverse path setup* (reverse route) is in the routing table. *rt0* depicts the reverse route and the process that the pseudocode checks to ensure a valid route exists. If the route is not in route table, an entry is created for the reverse route [8, 14, 15, 16 and 17].

```
{ rt_entry Xrt0;
Packet Xbuffered_pkt;
rt0 = rtable.rt_lookup(rq->rq_src);
if(rt0 == 0) {
rt0 = rtable.rt_add(rq->rq_src);}
if (rt0->rt_flags != RTF_UP) {
if (rt0->rt_req_timeout > 0.0) {
rt0->rt_req_cnt = 0;
rt0->rt_req_timeout = 0.0;
if (rq->rq_hop_count != INFINITY2)
rt0->rt_req_last_ttl = rq->rq_hop_count;
rt0->rt_expire = CURRENT_TIME + ACTIVE_ROUTE_TIMEOUT;}
else {rt0->rt_expire = CURRENT_TIME + REV_ROUTE_LIFE;}
```

Figure 10. Portion of NS2 Pseudocode for the Reverse Path Setup.

2. Forward Path Setup

The RREQ will eventually arrive at a node that contains the most current route to the destination. The RREQ uses the *source sequence number* and the *destination sequence number* for particular functions as previously stated in the *reverse path setup*. Intermediate nodes in the network function as beacons in determining the most current route to the destination. If an intermediate node receives a RREQ, it determines whether the route is current by checking the *destination sequence number* in the RREQ and comparing it to its own routing information concerning the destination node. The

17

intermediate node can reply to the RREQ when it contains a route with a *destination sequence number* that is greater than or equal to the *destination sequence number* in the RREQ. Otherwise, the intermediate node rebroadcasts the RREQ. The intermediate node can unicast a *route reply* (RREP) packet to the neighbor from which it received the RREQ under two conditions, both of which must be satisfied. If the intermediate node contains a current route to the destination node and the RREQ has not been processed previously, then the RREP can be sent. Figure 11 is a portion of the C++ pseudocode used for *forward path setup* resident within the Network Simulator 2 (NS2). The pseudocode describes the process of creating the forward path setup to ensure that a valid route exists. The RREP contains five key pieces of information: *source address,*

```
if ((rt->rt_flags != RTF_UP)
    ((rt->rt_seqno <rp->rp_dst_seqno) ||
    ((rt->rt_seqno ==rp->rp_dst_seqno) &&
    (rt->rt_hops >rp->rp_hop_count)))) {
    rt->rt_expire = CURRENT_TIME + rp->rp_lifetime;
        if (rt->rt_flags != RTF_UP)
        rt->error_propagate_counter = 0;
        rt->rt_flags = RTF_UP;
        rt->rt_hops =rp->rp_hop_count;
        rt->rt_seqno =rp->rp_dst_seqno;
        rt->rt_nexthop =ch->prev_hop_;
            rt->rt_errors = 0;
            rt->rt_error_time = 0.0;
    if ( ih->daddr() == index) { rt->rt_disc_latency[rt->hist_indx] =
            (CURRENT_TIME - rp->rp_timestamp)
            / (double)rp->rp_hop_count;
    rt ->hist_indx = ( rt->hist_indx + 1) % MAX_HISTORY;}
```

Figure 11. Portion of NS2 Pseudocode for the Forward Path Setup.

destination address, destination sequence number, hop count and *lifetime.* As the RREP traverses the network back to the source, two processes occur. The intermediate nodes along the path use the *reverse path setup* to forward the RREP, and forward links (*forward path setup*) are established when the RREP travels along the reverse path. As the RREP traverses intermediate nodes, each node updates its *route request expiration timer* information and records the most recent *destination sequence number* for the destination node. The *route request expiration timer* information is used to remove

18

reverse path route entries for intermediate nodes that are not on the path from the source to the destination node. This parameter depends on the actual size of the MANET. Figure 12 depicts the *forward path setup* from the source node S to the destination node D. Intermediate nodes that are not on the path, as determined by the RREP, will

Represents a link on the forward path
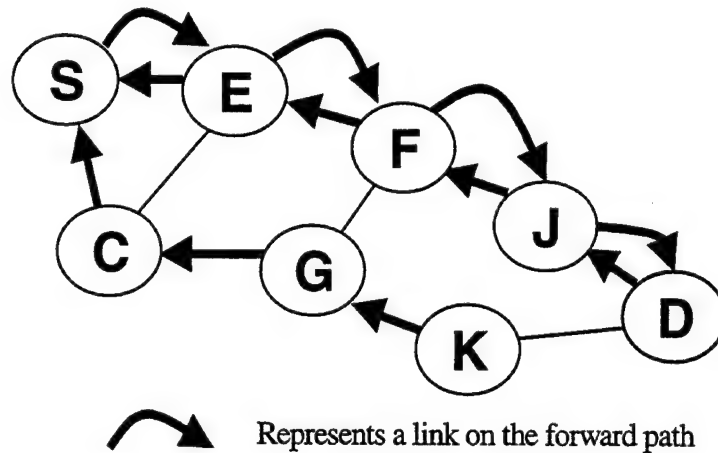
Figure 12. Forward Path Setup from Source S to Destination D.

automatically timeout and delete the reverse pointers. If an intermediate node receives a RREP with a better metric, it updates its route entry and propagates this RREP towards the source node. If an intermediate node receives a RREP without a better metric, it suppresses the RREP and deletes it with no further propagation. The source node can begin transmitting data once it receives the first RREP. Additionally, the source node can update its routing information if it learns of a better route at any time [8, 14, 15, 16 and 17].

B.   ROUTE TABLE MANAGEMENT

Route table management determines whether a route is still active using four primary parameters: *source sequence numbers, destination sequence numbers, route request expiration timer* and *route caching timeout.* The first three parameters have previously been defined. The *route caching timeout* is the time beyond which a route is no longer considered to be valid. Each mobile node contains a route table entry with the

19

following information: *destination, next hop, number of hops, sequence number for the destination, active neighbors for this route* and *the expiration time for the route table entry*. With this information, each node can determine whether its neighbor is considered active for the particular destination. The criterion for being active is determined if the neighbor originates or relays at least one packet for a destination within the most recent *active timeout* period. This enables all active source nodes to become informed if a link along a path to the destination fails or breaks. Each time a route entry is used to transmit data, the *expiration time* is updated to the current time plus the *active route timeout*. Figure 13 is a simple illustration of four nodes communicating together and the associated routing table each node maintains after the route discovery process. Route entries may be updated if a route with a greater *destination sequence number* or a smaller *hopcount* (number of hops) to the destination node is discovered [8, 14, 15, 16 and 17].
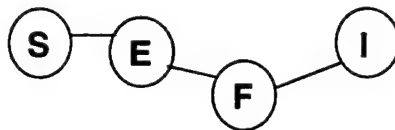
Node S		Node E		Node F		Node I	
Destination	Nexthop	Destination	Nexthop	Destination	Nexthop	Destination	Nexthop
E	E	S	S	E	E	F	F
I	E	F	F	I	I		
		I	F	S	E		

Figure 13. Four Node Scenario with Routing Table.

20

C. PATH MAINTENANCE

Due to the movement of nodes throughout the network, path maintenance is used to ensure that routes from the source to the specified destination are still valid. Prior to performing path maintenance, AODV follows a specified criteria. The movement of a node, or nodes, not along the active path, does not trigger path maintenance since these nodes do not affect the routing to the specified destination. The movement of the source node does not trigger path maintenance since the source node can reinitiate route discovery to establish a new route to the specified destination. The movement of the specified destination or an intermediate node does trigger path maintenance. When any of these nodes move, nodes upstream of the break propagate *unsolicited* RREPs to all active upstream neighbors. The information provided within the *unsolicited* RREP includes a fresh sequence number, one different from the previously known sequence number, and a hop count of infinity. The nodes propagate the *unsolicited* RREP until all the active sources receive the *unsolicited* RREP. The *unsolicited* RREP terminates because AODV maintains only loop-free routes and the hop count of infinity violates the number of nodes in the MANET.

The source node can reinitiate route discovery if a route to the destination is still required. The source node propagates a new RREQ with a *destination sequence number* of one greater than the last known *destination sequence number*. The new RREQ with a new *destination sequence number* is required to ensure that a new route is created and that nodes will not reply if they still regard the previous information valid to the specified destination. Figure 14 provides an illustration on path maintenance. The initial route shows node J moving to a new location J'. Node F, upstream of node J, notices the loss of the link and sends an *unsolicited* RREP to node E. Node E forwards this *unsolicited* RREP to the source node S. The source node S reinitiates route discovery and finds a new route through node K to the destination node D. Figure 15 is a portion of the C++ pseudocode used for path maintenance (*unsolicited* RREP) resident within NS2. The pseudocode describes the process of broadcasting an *unsolicited* RREP to the upstream neighbors. In this case, the pseudocode must also include purging the network interface

21

| The Initial Route | After Route Reconstruction |
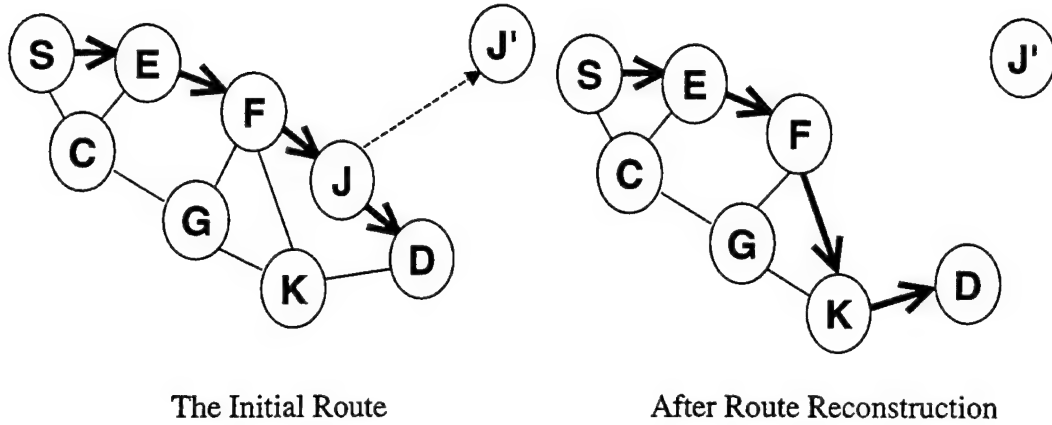
Figure 14. Path Maintenance Setup from Source S to Destination D.

queues that may have packets destined for this broken neighbor. Once the upstream neighbors are notified, the source node is informed of the break. If a node is the source of the packet, it will queue this packet and send a RREQ. Otherwise, the node will drop the packet: nothing is salvaged by an intermediate node [8, 14, 15, 16 and 17].

```
sendTriggeredReply(rt->rt_dst, rt->rt_seqno);
                    #endif
                       {
                   Packet Xp;
while((p = ifqueue->filter(rt->rt_nexthop))) {
          struct hdr_ip Xih = HDR_IP(p);
             if (index == ih->saddr()) {
                 rqueue.enque(p);
              sendRequest(ih->daddr());
                       }
                    else {
        drop(p, DROP_RTR_NO_ROUTE);
```

Figure 15. Portion of NS2 Pseudocode for Path Maintenance (*unsolicited* RREP).

D. LOCAL CONNECTIVITY MANAGEMENT

Nodes use the local connectivity management to learn who their active neighbors are and to know if these neighbors are still in range of communication. There are two methods by which a node can gather information concerning its active neighbor, reception of a broadcast or a *hello* message. A broadcast message, RREP or RREQ, is

22

propagated throughout the system in determining a route to the specified destination. Nodes that receive this broadcast message update their local connectivity information to include this new neighbor. A node that has not sent any packets to all of its active downstream neighbors within a *hello_interval* propagates a *hello* message. A *hello* message is a special *unsolicited* RREP and contains information concerning its identity and *sequence number*. A node that receives a *hello* message does not update or change its own *sequence number*; rather the neighboring nodes update their local connectivity information to include this new neighbor. Figure 16 is a portion of the C++ pseudocode used for local connectivity (adding new neighbors) resident within NS2. The pseudocode describes the process that the node will use to check its current list of neighbors and to update its list of neighbors when a change of neighbors is discovered. Either a new neighbor is added to the list, or a known neighbor is no longer reachable and subsequently deleted from the list. Its immediate neighbors only receive the hello message, and the

```
AODV::nb_insert(nsaddr_t id)
{Neighbor Xnb = new Neighbor(id);
            assert(nb);
nb->nb_expire = CURRENT_TIME +
(1.5 X ALLOWED_HELLO_LOSS X HELLO_INTERVAL);
LIST_INSERT_HEAD(&nbhead, nb, nb_link);
            seqno += 1;}
            NeighborX
AODV::nb_lookup(nsaddr_t id)
{Neighbor Xnb = nbhead.lh_first;
for(; nb; nb = nb->nb_link.le_next) {
    if(nb->nb_addr == id)
            break;}
        return nb
```

Figure 16. Portion of NS2 Pseudocode for Local Connectivity.

hello message is not propagated throughout the network because its time to live (TTL) value is set to one. Nodes verify that routes are used only from neighbors that have received the node's *hello* message. Since nodes must hear from active neighbors

periodically, failure to receive a *hello* message indicates a node is out of range and removed from the node's local connectivity. [8, 14, 15, 16 and 17].

E. SUMMARY

Chapter III has presented the AODV protocol and the procedures through which it establishes a route, maintains a route, adjusts to the movement of nodes into and out of the network, and manages the routing table and the local connectivity. AODV is a hybrid of DSR and DSDV. From DSR, AODV incorporates a broadcast discovery mechanism, and from DSDV it incorporates the most recent routing information between nodes. AODV minimizes the network load for control and data traffic, is responsive to topology changes, is capable of adjusting to changes in topology, and ensures loop-free routing even while repairing broken links.

IV. SIMULATION

The simulation software used in this thesis was NS2, Version 2.1b6 on a Linux platform. Perkins and Royer's AODV NS2 implementation, originally developed in a Parallel Simulation Environment for Complex Systems (PARSEC) environment and later converted to work in UNIX, LINUX and Windows environments, was one of four MANET protocols available at the time of this work. The other three MANET protocols available are DSR, DSDV and the Temporally Ordered Routing Algorithm (TORA). The NS2 package was chosen for this MANET protocol research due to its availability as freeware on the Internet from the University of California (UCB) at Berkeley and because of the numerous MANET routing protocols available for evaluation. Other popular simulation software packages used for MANET protocol simulation include Optimum Network Performance (OPNET), PARSEC, and the C++ programming language.

A. NETWORK SIMULATOR 2 (NS2)

NS2 version 2.1b6 was created by the Virtual InterNetwork Testbed (VINT) project funded by the Defense Advanced Research Projects Agency (DARPA). The VINT project is a collaborative project that includes the University of California (UCB) at Berkeley, University Southern California (USC)/Information Sciences Institute (ISI), Xerox Palo Alto Research Center (PARC) and the Lawrence Berkeley National Laboratory (LBNL). The purpose of this project is to build a network simulator that will allow the study of scale and protocol interaction in the context of current and future network protocols. The simulator is written in the C++ programming language and the Object Tool Command Language (OTCL). The user writes an OTCL script that defines the mobile nodes in the network, the traffic in the network and the routing protocol. The Tool Command Language (TCL) script is also created by the user that contains the files generated in OTCL and called by the simulator. Appendix A contains one example of the TCL script files generated by the user, and Appendix B contains an output trace file used in the simulation. The OTCL script is used by NS2 to perform the simulation. The result

of the simulations is an output trace file that is used for further data processing (parsing) and to visualize the simulation with a resident program tool in NS2 called the Network Animator (NAM).

Figure 17 depicts an overview of how a simulation is performed in NS2 from the user input, in the OTCL script, to data processing [18]. The user creates node movement and traffic generation files. A TCL script is used to bridge the OTCL script created by the user with the C++ code resident in the NS2 simulator to perform the simulation. The NS2 simulator performs the simulation and creates an output file containing results of the simulation. The user can add the network animator (NAM) to the TCL script to view the movement of MANET nodes in the simulation. The output trace file is parsed using a perl script, and the results of the data processing are analyzed in MATLAB.
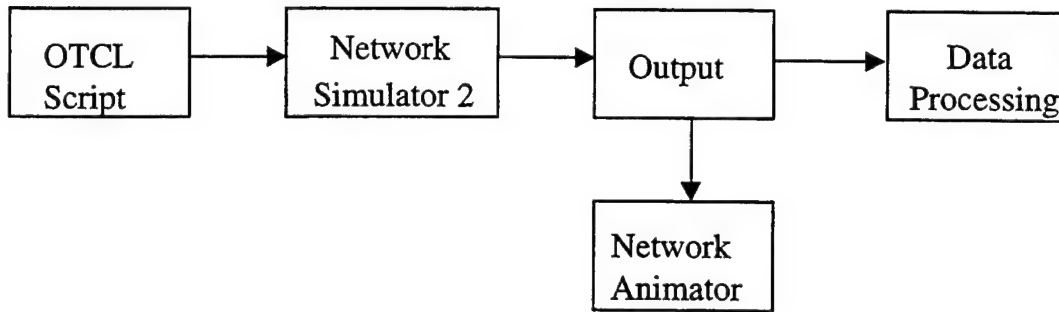
Figure 17. Overview of NS2.

B.   AODV MODEL

The AODV NS2 model contains two different levels of software programming. The user level in OTCL, and the specific AODV model resident in the simulator in the C++ programming language. The user level in OTCL is implemented by generating a mobile node movement file, generating a traffic pattern file and designating a routing protocol. Each specific user file generation will be discussed in further detail in the following sections. The user has flexibility in changing various parameters for each simulation in NS2. For node movement, these variables include: the number of nodes in the network (–n), the pause time in between node movements (–p), the maximum speed

26

of each node in meters per second (–s), the total simulation time in seconds (–t), the boundary in the x axis in meters (–x) and the boundary in the y axis in meters (–y). For the traffic pattern, these variables include: the type of traffic (–type, Transmission Control Protocol (TCP) or Constant Bit Rate (CBR)), total number of nodes in the network (–nn), the maximum number of connections set up between them in the network (–mc) and the rate of packet distribution in packets per second (-rate).

Figure 18 depicts the AODV design for NS2 in the two software programming levels. The functions below the OTCL level are resident within NS2 and are written in
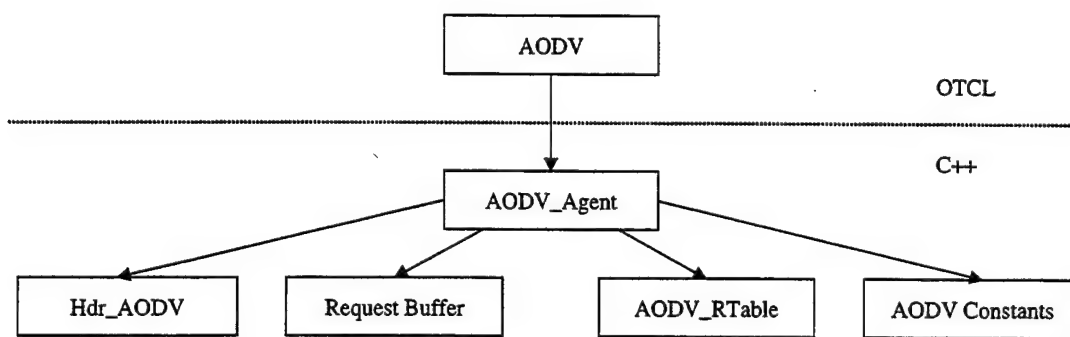
Figure 18. AODV Design of Implementation for NS2.

the C++ programming language. The functions at this level are the default parameters of the AODV model itself and are not normally modified by the user. The major functions are the *AODV_agent, Hdr_AODV, Request Buffer, AODV_Rtable* and *AODV constants.* The *AODV_agent* handles the various RREQ, RREP, *hello,* and *unsolicited* RREP messages. It also has a send buffer that buffers packets during route discovery, and the timers that maintain timeouts for route entries and the send buffer are implemented in this function. The AODV header (*Hdr_AODV*) defines the specific message format for the various messages in AODV. The *Request Buffer* ensures a node does not process the same RREQ numerous times and stores processed requests. The AODV route table (*AODV_Rtable*) maintains and updates the routes and implements the active neighbor list for each route entry. The *AODV constants* contains all the defined constants within the AODV model that include: *hello interval, active route timeout, route reply lifetime,*

27

allowed hello loss, request retries, time between retransmitted requests, time to hold packets awaiting routes and *maximum rate of sending replies for a route* [17].

1. Mobile Node Movement

Each mobile node movement makes use of a routing agent for the purpose of calculating routes to other nodes within the MANET. Figure 19 depicts the mobile node mechanism and implementation in NS2. Packets are sent from the application and are received by the routing agent. The agent determines a path that the packet must travel to reach the destination and stamps it with this information. The packet is then sent down to the link layer. The link layer uses an Address Resolution Protocol (ARP) to determine the hardware addresses of neighboring nodes and map IP addresses to the correct interfaces for dissemination. When the information is known, the packet is sent down to the interface queue and awaits a signal from the Medium Access Control (MAC) protocol. When the MAC layer determines that it can send a packet onto the channel, it grabs the packet from the queue and moves it over to the network interface. The network interface sends the packet onto the radio channel. The packet is copied and delivered to all network interfaces at the time that the first bit of the packet would begin arriving at the interface in a physical system. Each network interface stamps the packet with the receiving interface information and then invokes the propagation model. The propagation model uses the transmit and receive stamps to determine the receive power for the interface.

The above procedure is then reversed on the receiving network. This process happens within NS2 as information is being disseminated. As previously mentioned, the user creates a specific node movement with certain parameters. These parameters include: the number of nodes in the network (–*n*), the pause time in between node movements (–*p*), the maximum speed of each node in meters per second (–*s*), the total simulation time in seconds (–*t*), the boundary in the x axis in meters (–*x*) and the boundary in the y axis in meters (–*y*). The shaded line below depicts the command line input for the generation of mobile node movement file in NS2 with an associated file for

28

this input created by the user. Appendix C contains a node movement file generated by the user in the simulation [17].

./setdest –n 50 –p 500 –s 1.78 –t 600 –x 1500 –y 300 > scen50-vel1.78-ty
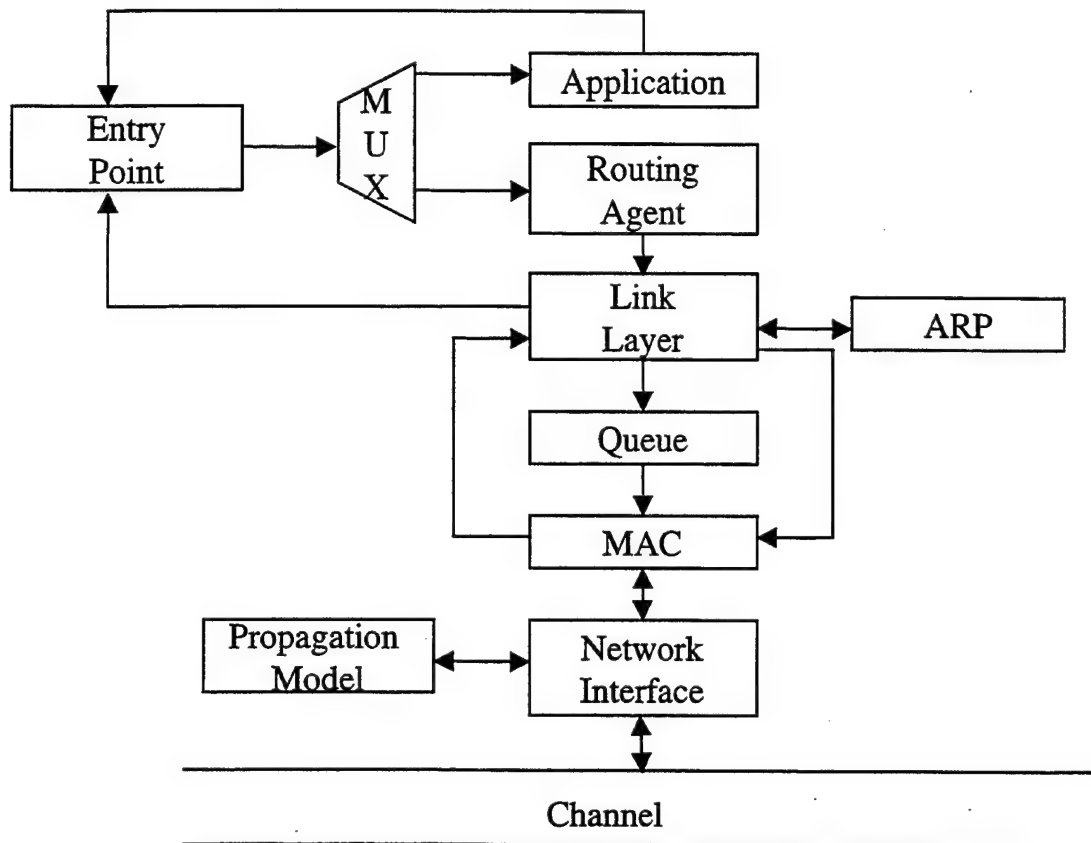
Figure 19. A Mobile Node in NS2 (From Ref. [17]).

2. Traffic Pattern Generation

The creation of the random traffic pattern can be established between mobile nodes using a traffic scenario generator script. Either constant bit rate (CBR) or transmission control protocol (TCP) traffic connections can be created. The models used in this thesis contained only CBR connections. A generic CBR generation file is resident (cbrgen.tcl) in NS2 and is called by the user in the creation of a user specific traffic pattern file. Each traffic pattern file generated is unique. The user is able to generate a specific traffic connection using the CBR generation file and set certain variables. These

29

variables include: the type of traffic (*–type,* TCP or CBR), total number of nodes in the network (*–nn*), the maximum number of connections setup between them in the network (*–mc*) and the rate of packet distribution in packets per second (*-rate*). The shaded line below depicts the command line input for the generation of the CBR traffic pattern in NS2 with an associated file for this input created by the user. Appendix D contains a traffic pattern file generated by the user in the simulation [17].

ns cbrgen.tcl –type cbr –nn 50 –seed 1.0 –mc 20 –rate 4.0 > cbr-50-rate4-ty

3. Statistics Production

The completion of all simulations in NS2 generates an output trace file. NS2 has no resident statistic production capability as is available in OPNET. Each output trace file is parsed using either an *awk* or perl script command to collect specific data from the output file for analysis. The output trace files generated by the simulation can exceed several gigabytes of storage capacity; therefore, only a maximum of 20-30 active connections were used. For example, the calculation for each statistic, packet fraction delay, throughput, etc., is parsed from the output file using a perl script then dumped into MATLAB to organize and produce results for analysis. Appendix B contains an output trace file generated in the simulation.

C. SUMMARY

This chapter has provided an introduction of the implementation of NS2 version 2.1b6 and has presented the AODV model. NS2 maintains resident features of the AODV model in C++, and the user is able to generate a specific node movement, traffic pattern and routing protocol in the OTCL/TCL script. The mobile nodes are modeled to work in a MANET environment and are executed through a mobile node mechanism as depicted in Figure 19. The user has the ability to change numerous parameters for the mobile node movement and the traffic pattern. Statistics were collected on each simulation through parsing of the output file then analyzed in MATLAB.

V. RESULTS

In this chapter, the results of the AODV NS2 simulations are presented. The focus is to use parameters established by Corson and Macker in RFC 2501 for the evaluation of AODV. The goodput metric was also included as another indicator of the performance of AODV since the measurement of actual packets received by the destination node from the source node is evaluated. The simulation performance of AODV is reported in terms of packet fraction delivery, route loss, buffer loss, total loss, throughout, and goodput in a 50 node MANET with a maximum of 20 connections. One simulation contained 100 nodes with a maximum of 30 connections for the 2500m × 2500m network environment. Various network environment sizes, pause times, velocities and packet rates were used during the simulation and will be explained in further detail later in this chapter. Section A provides an explanation of the scenario and configuration development. Section B provides the various metrics generated by AODV in the simulation with an explanation of the calculation and results. Section C is a summary of the results.

A.   SCENARIO

The network configuration used in this scenario is typical to the tactical employment of the JTRS by U.S. armed forces. The network implementation was maintained at 50 nodes with a maximum of 20 connections for the 500m × 500m and the 1500m × 1500m network environments. One network implementation contained 100 nodes with a maximum number of 30 connections for the 2500m × 2500m network environments. A larger number of connections could be used; however, the processing time and available space of the computer hard drive were limiting factors. The default parameters used in the simulations are listed in Table 1.

Parameter	Value
Transmitter range	250 meters (m)
Simulation time	600 seconds (s)
Number of Nodes	50, 100
Pause time	0, 20, 50, 120, 300, 500, 600 s
Environment Size	500m × 500m, 1500m × 1500m, 2500m × 2500m
Traffic Type	Constant Bit Rate (CBR)
Packet Rate	4, 16, 30 packet/sescond (p/s)
Packet Size	64 bytes
Transmitter Power	2.818 milliwatts (mW)
Velocity	1.78, 11.17, 33.5 m/s

Table 1. Parameters Used During NS2 Simulations.

1. Configuration

Each simulation was configured using the same parameters listed in Table 1, except for adjustments to network environment sizes, pause times, velocities and packet rates. Three different network environment sizes were chosen to provide a realistic view of the performance of AODV according to size. A small, 500m × 500m, environment and a large, 1500m × 1500m, environment were chosen to reflect different battlespace sizes. Seven different pause times were used to represent a high mobility to a low mobility environment. In terms of mobility, a wide range was chosen to reflect most environments encountered. Three different velocities were chosen: 1.78 m/s, 11.17 m/s and 33.5 m/s, representative of a foot mobile US Marine, a tactical/commercial vehicle and a low speed helicopter, respectively. NS2 uses these values to represent the maximum velocity a node can move, and randomly moves a node in the interval 0 to 1.78 m/s, 0 to 11.17 m/s or 0 to 33.5 m/s as applicable. Constant bit rate (CBR) traffic was used in each simulation. However, three different packet rates were chosen to reflect voice (4 p/s), data (16 p/s) and video teleconferencing (VTC, 30 p/s) traffic. Note that data is typically TCP traffic, but is treated as CBR traffic in this work.

The results of a simulation are stored as an output trace file. The trace file is parsed using a perl script to extract the required data. Each simulation is performed in

NS2 using specified parameters. The first set of simulations contained a network environment of 500m × 500m, velocities of 1.78 m/s, 11.17 m/s and 33.5 m/s, and packet rates of 4, 16, and 30 p/s. A total of 9 simulations were run using the network environment of 500m × 500m. All other parameters remained the same as listed in Table 1. The second set of simulations contained a network environment of 1500m × 1500m, velocities of 1.78 m/s, 11.17 m/s and 33.5 m/s, and packet rates of 4, 16, and 30 p/s. A total of 9 simulations were run using the network environment of 1500m × 1500m. All other parameters remained the same as listed in Table 1. For example, the first simulation contained a network environment of 500m × 500m, a velocity of 1.78 m/s, a packet rate of 4 p/s, the seven pause times and all other parameters as listed in Table 1.

B. PERFORMANCE

The performance metrics of various MANET parameters were analyzed. Six key performance metrics were evaluated: packet delivery fraction, route loss, buffer loss, total losses, throughput and goodput. Furthermore, the performance metrics for each simulation were repeated a total of 12 times and averaged together for each data point (Monte Carlo simulation). These simulations were performed for the seven pause times for a total of 84 simulations per simulation set. For all the scenarios a grand total of 2268 simulation runs were performed. Each performance metric will be discussed in detail concerning the generation and collection of data for these various metrics.

1. Packet Delivery Fraction

The *packet delivery fraction* is measured as a ratio of the total number of data packets received by the destination node to the number of data packets sent by the source node. Data packets may be dropped en route to the destination for two reasons: the next hop link is broken when the data packet is ready to be transmitted; or no routing table entry information exists for the requested destination. The size of the network and the velocity of the nodes significantly affect this metric. Figures 20 and 21 show plots of the packet delivery fraction as a function of pause time for a network environment of 500m ×

33

500m and 1500m × 1500m, velocities of 1.78 m/s, 11.17 m/s and 33.5 m/s, and packet rates of 4, 16, and 30 p/s. All other parameters remained the same as listed in Table 1.

Figure 22 represents the packet delivery fraction in a large network environment of 2500m × 2500m with 100 nodes, 30 connections, a fixed velocity of 11.17 m/s and a fixed packet rate of 16 p/s. The results of this large simulation are compared to the 500m × 500m and 1500 × 1500m network environments with 50 nodes and 20 connections, a fixed velocity of 11.17 m/s and a fixed packet rate of 16 p/s. All other parameters remained the same as listed in Table 1. The 2500m × 2500m network environment was simulated to test the limitations of the routing protocol in terms of packet delivery fraction and to evaluate the capability of the routing protocol with a large number of mobile nodes and varied network environments.

Figure 23 represents the packet delivery fraction of a 500m × 500m network environment with a varied offered load. The offered load is represented by the rate at which packets are being sent by the source node. Packet rates of 1, 2, 3, 4, 10, 16 and 20 p/s were used. These packet rates correspond with the following offered load in kilo bits per second (kbps): 81.920, 163.840, 245.760, 327.680, 819.200, 1310.720 and 1638.400, respectively.

The simulations yielded varied results. The results of the 500m × 500m network environment (see Figure 20) demonstrate that different packet rates yield a different packet delivery fraction, regardless of the velocity of the nodes. The different packet delivery fractions are due to the relatively small network environment. Average packet delivery fractions of 100%, 55% and 30% are obtained for packet rates of 4 p/s, 16 p/s and 30 p/s, respectively.

The 1500m × 1500m network environment (see Figure 21) returned results similar to the 500m × 500m network environment, but with a lower packet delivery fraction due to the larger network environment. Average packet delivery fractions of 50%, 30% and 18% are obtained for packet rates of 4 p/s, 16 p/s and 30 p/s, respectively.

The 2500m × 2500m network environment (see Figure 22) produced results similar to the 500m × 500m and the 1500m × 1500m network environments, but with a

34

lower packet delivery fraction due to the larger network environment. The comparison in Figure 23 represents three different network environments with a fixed rate of 16 p/s and a fixed velocity of 11.17 m/s. The smallest network environment of 500m × 500m yielded the best packet delivery fraction of 52% while the larger network environments of 1500m × 1500m and 2500m × 2500m yielded packet delivery fraction of 26% and 14%, respectively. Packet delivery fraction as a function of the offered load for the 500m × 500m network environment is shown in Figure 23. Small offered loads below 327 kbps yielded a packet delivery fraction of almost 100% while offered loads over 1,638 kbps yielded a packet delivery fraction of 49%.
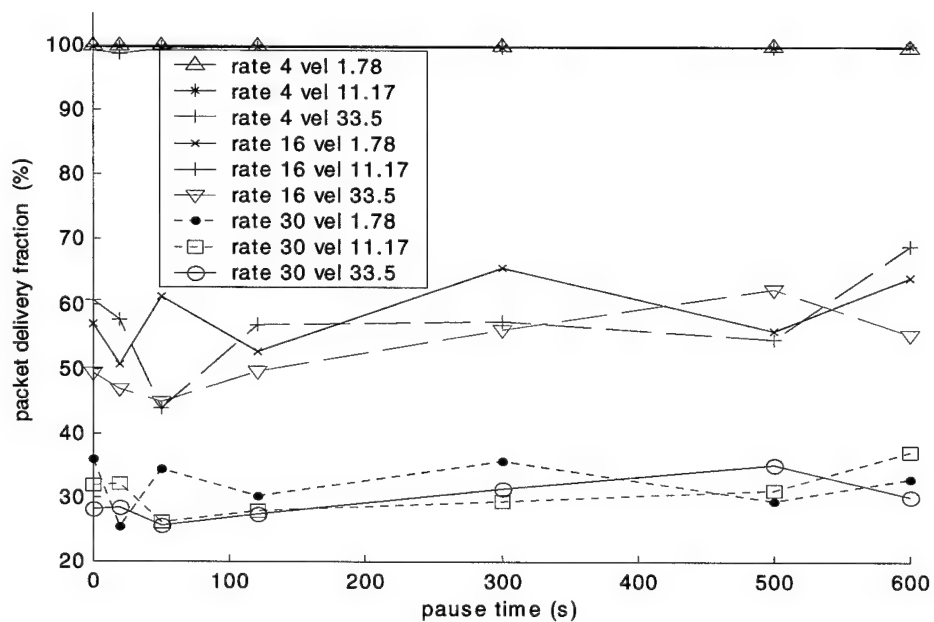
Figure 20. Packet Delivery Fraction for a Network Environment of 500m × 500m with Varied Packet Rates and Node Velocities.

35
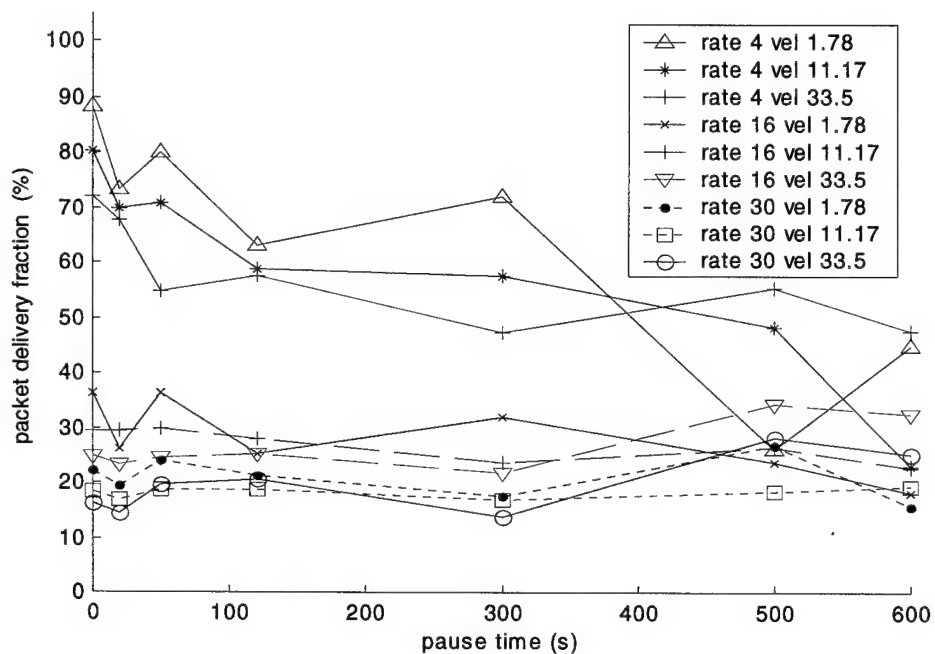
Figure 21. Packet Delivery Fraction for a Network Environment of 1500m × 1500m with Varied Packet Rates and Node Velocities.

Figure 22. Packet Delivery Fraction for Varied Network Environments with a Fixed Packet Rate of 16 p/s and a Fixed Velocity of 11.17 m/s.

36
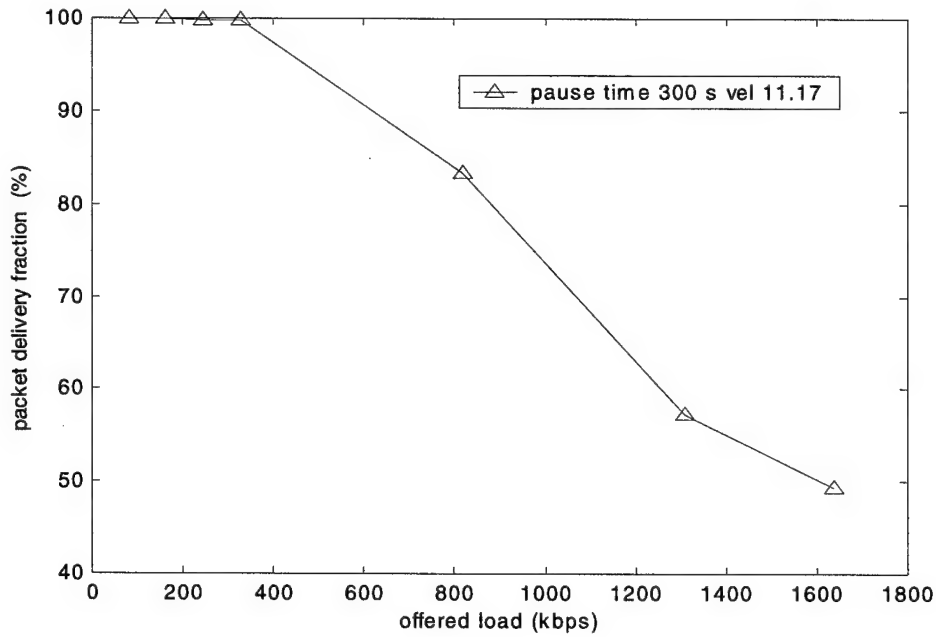
Figure 23. Packet Delivery Fraction for a Network Environment of 500m × 500m with a Fixed Pause Time (300 s) and Velocity (11.17 m/s) Having a Varied Packet Rate/Offered Load.

2. Routing Loss

The *routing loss* is measured as the number of packets that are dropped from the source to the destination. The principal reason for routing loss is due to the random movement of the nodes, which causes a continuous change in the network topology. A route that was previously established and that was forwarding packets may have moved out of range of an intermediate node, thus forcing the data packets to be dropped. In this case, the source will initiate a new route discovery to reestablish a connection; however, data packets are lost in the process. Figures 24 and 25 present plots of the routing loss for network environments of 500m × 500m and 1500m × 1500m, velocities of 1.78 m/s, 11.17 m/s and 33.5 m/s, and packet rates of 4, 16, and 30 p/s. All other parameters remained the same as listed in Table 1.

Route loss for a network environment of 2500m × 2500m with 100 nodes, 30 connections, a fixed velocity of 11.17 m/s and a fixed packet rate of 16 p/s is shown in Figure 26. The results of this simulation are compared to the 500m × 500m and 1500m ×

37

1500m network environments with 50 nodes and 20 connections, a fixed velocity of 11.17 m/s and a fixed packet rate of 16 p/s. All other parameters remained the same as listed in Table 1.

The simulations yielded varied results. The 500m × 500m network environment (see Figure 24) results demonstrate that different packet rates yield a different routing loss, regardless of the velocity of the nodes, due to the relatively small network environment. Average route losses of 10, 400 and 1000 packets are obtained for packet rate of 4 p/s, 16 p/s and 30 p/s, respectively.

The 1500m × 1500m network environment (see Figure 25) returned similar results as the 500m × 500m network environment, but with a larger routing loss due to the larger network environment. Average route losses of 900, 4300 and 6600 packets are obtained for packet rates of 4 p/s, 16 p/s and 30 p/s, respectively.

The 2500m × 2500m network environment (see Figure 26) produced results similar to the 500m × 500m and the 1500m × 1500m network environments, but with a larger routing loss due to the larger network environment and changing topology causing packets to be dropped. The comparison in Figure 26 represents three different network environments with a fixed rate of 16 p/s and a fixed velocity of 11.17 m/s. The smallest network environment of 500m × 500m yielded the least route loss with an average routing loss of 400 packets while the larger network environments of 1500m × 1500m and 2500m × 2500m yielded an average routing loss of 4000 and 6500 packets, respectively.

In general, an increase in the size of the network environment increased the amount of route loss. Additionally, an increase in the packet rate contributed to the amount of route loss. In the 1500m × 1500m network environment (see Figure 25), the route loss for packet rates of 16 and 30 p/s were approximately the same.

Figure 24. Routing Loss for a Network Environment of 500m × 500m with Varied Packet Rates and Node Velocities.

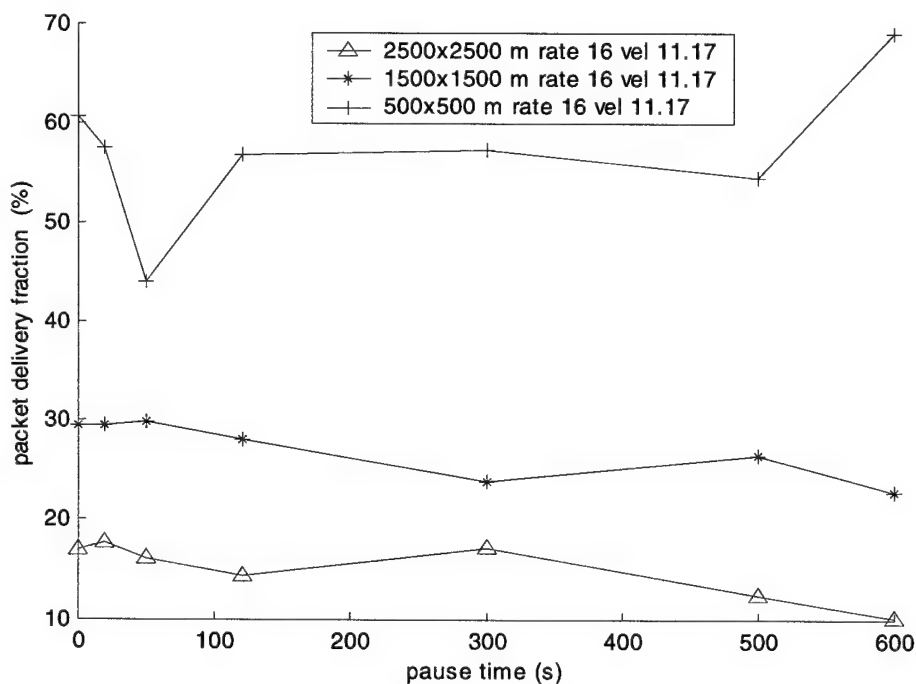Figure 25. Routing Loss for a Network Environment of 1500m × 1500m with Varied Packet Rates and Node Velocities.

39

Figure 26. Routing Loss for Varied Network Environments with a Fixed Packet
Rate of 16 p/s and a Fixed Velocity of 11.17 m/s.

3. Buffer Loss

The *buffer (queue) loss* is a consequence of the overflow of packets in each node's queue. This occurs as the data packets are held in the queue while waiting to be forwarded to the next hop node. This loss can be affected significantly by the movement and subsequent loss of communication with an intermediate node attempting to forward data packets. As the intermediate node's buffer is limited in size, packets begin to accumulate when a neighbor node cannot be located. Packets continue to be received, and eventually the buffer is filled to capacity. As a result data packets are dropped. Figures 27 and 28 illustrate the buffer loss for a network environment of 500m × 500m and 1500m × 1500m, velocities of 1.78 m/s, 11.17 m/s and 33.5 m/s, and packet rates of 4, 16, and 30 p/s. All other parameters remained the same as listed in Table 1.

Figure 29 displays the buffer loss in a large network environment of 2500m × 2500m with 100 nodes, 30 connections, a fixed velocity of 11.17 m/s and a fixed packet rate of 16 p/s. The results of this large simulation are compared to the 500m × 500m and 1500m × 1500m network environments with 50 nodes and 20 connections, a fixed

40

velocity of 11.17 m/s and a fixed packet rate of 16 p/s. All other parameters remained the same as listed in Table 1. The 2500m × 2500m network environment was simulated to test the limitations of the routing protocol in terms of buffer loss and to evaluate the capability of the routing protocol with a large number of mobile nodes and varied network environments.

The simulations yielded varied results. Results of the 500m × 500m network environment (see Figure 27) demonstrate that the buffer losses are dependent on packet rates but not on the node velocities. Average buffer losses of 5, 3600 and 9800 packets are obtained for packet rates of 4 p/s, 16 p/s and 30 p/s, respectively. In this network environment, the majority of the nodes will be able to establish and maintain connections, but an increase in the packet rate saturates the buffers in the intermediate nodes, thereby causing loss. Larger network environments avoid this problem because connections are unable to be maintained as well as in a small network environment.

The 1500m × 1500m network environment (see Figure 28) returned results similar to the 500m × 500m network environment, but with a smaller buffer loss. Average buffer losses of 10, 1200 and 3500 packets are measured for packet rates of 4 p/s, 16 p/s and 30 p/s, respectively.

The 2500m × 2500m network environment (see Figure 29) produced smaller buffer losses due to the larger size. The comparison in Figure 29 represents three different network environments with a fixed rate of 16 p/s and a fixed velocity of 11.17 m/s. The smallest network environment of 500m × 500m yielded the largest average buffer loss of 3500 packets while the larger network environment of 1500m × 1500m and 2500m × 2500m yielded an average buffer loss of 800 and 250 packets, respectively.

In general, an increase in the size of the network environment decreased the amount of buffer loss (see Figures 28 and 29). Since routes are more difficult to maintain in larger network environments, the intermediate node buffers do not become saturated with packets. As a result, small network environments of 500m × 500m are most vulnerable to buffer loss (see Figure 27). Additionally, an increase in the packet rate contributed to the amount of buffer loss only in a small network environment of 500m ×

500m. In the 1500m × 1500m network environment (see Figure 28), the buffer loss for packet rates of 16 and 30 p/s were approximately the same values.
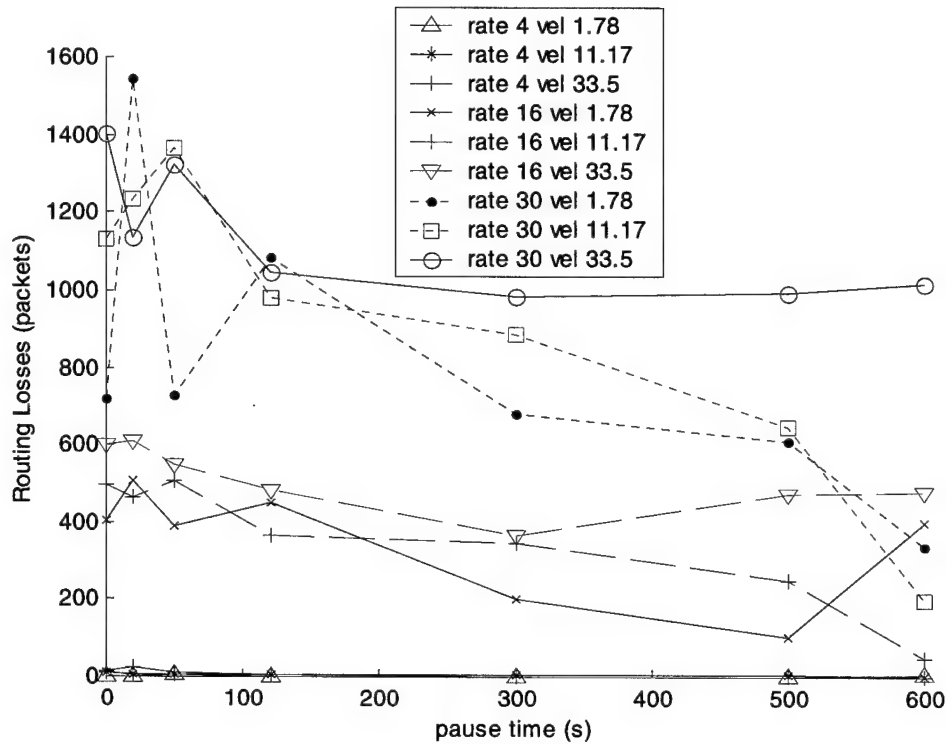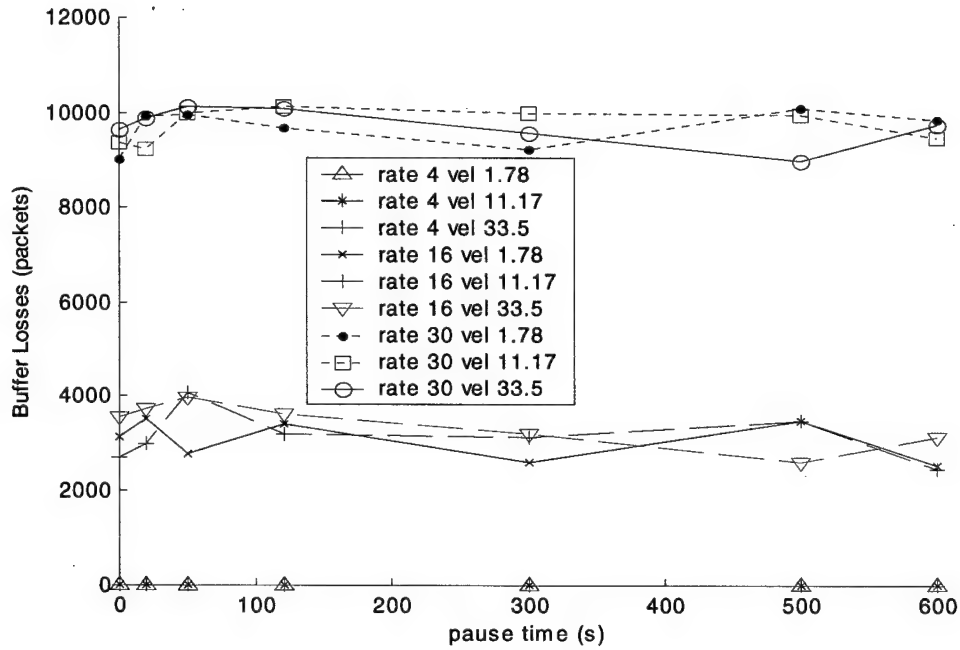

Figure 27. Buffer Loss for a Network Environment of 500m × 500m with Varied Packet Rates and Node Velocities.
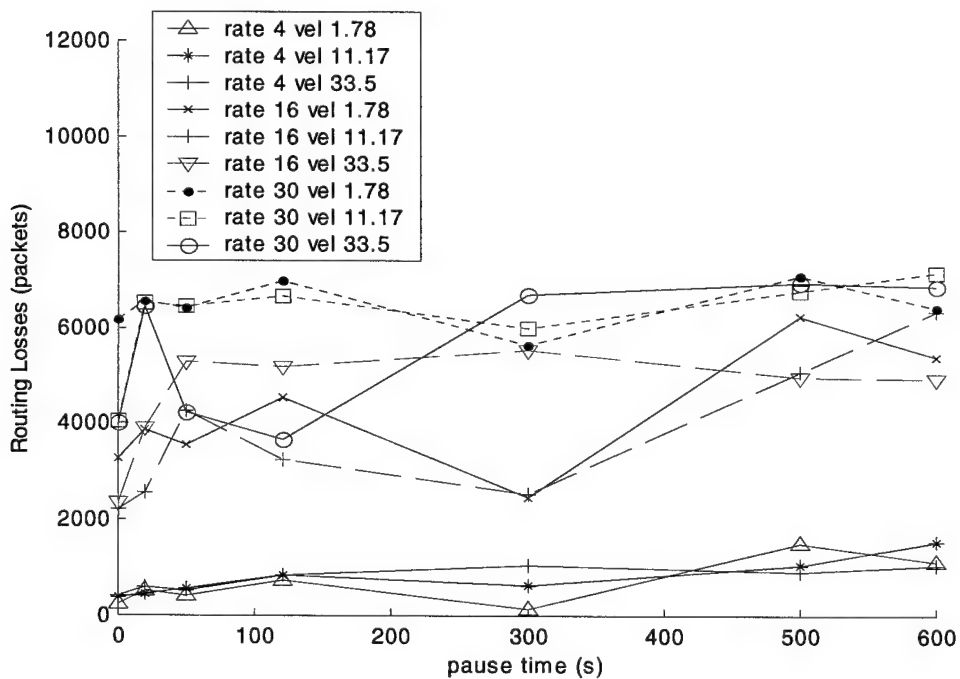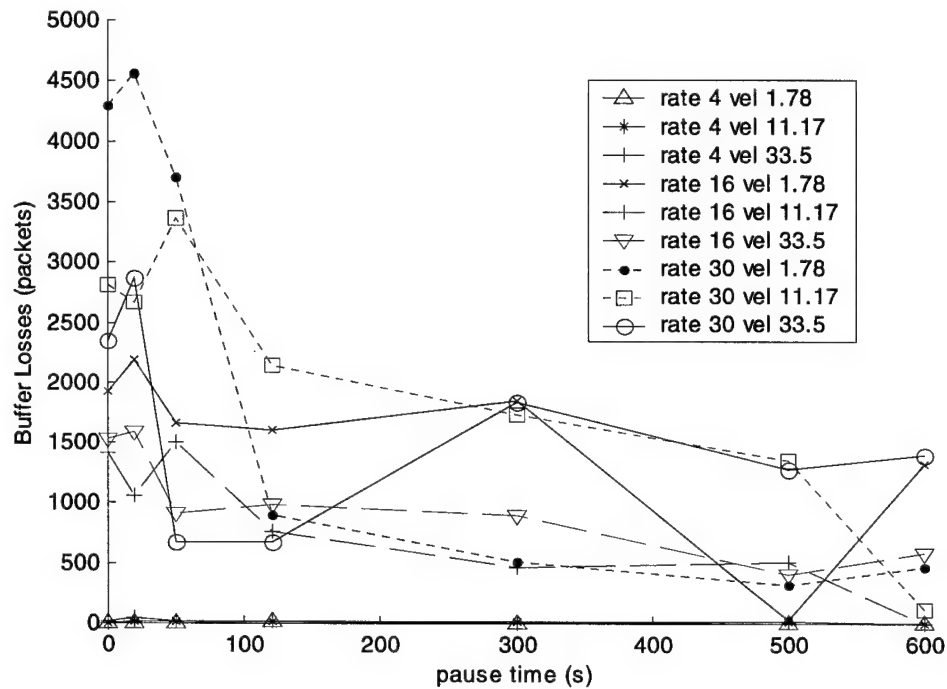
Figure 28. Buffer Loss for a Network Environment of 1500m × 1500m with Varied
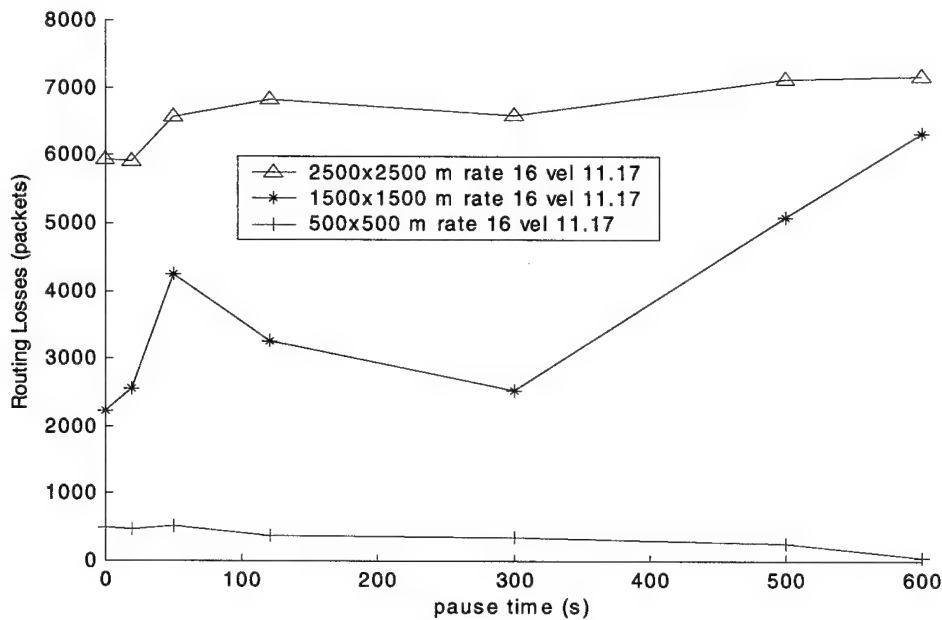Packet Rates and Node Velocities.

Figure 29. Buffer Loss for Varied Network Environments with a Fixed Packet Rate
of 16 p/s and a Fixed Velocity of 11.17 m/s.

43

4. Total Loss

The *total loss* is measured as the sum of the routing loss and the buffer loss. Figures 30 and 31 display the plots of total loss for a network environment of 500m × 500m and 1500m × 1500m, velocities of 1.78 m/s, 11.17 m/s and 33.5 m/s, and packet rates of 4, 16, and 30 p/s. All other parameters remained the same as listed in Table 1. The results of the 500m × 500m network environment produced (see Figure 30) average total losses of 20, 3800 and 11,000 packets for packet rates of 4 p/s, 16 p/s and 30 p/s, respectively. The 1500m × 1500m network environment (see Figure 31) returned similar results as the 500m × 500m network environment, but with a greater total loss due to the larger network environment. Average total losses of 1000, 5800 and 9500 packets are obtained for packet rates of 4 p/s, 16 p/s and 30 p/s, respectively.

Figure 32 represents the total loss in a large network environment of 2500m × 2500m with 100 nodes, 30 connections, a fixed velocity of 11.17 m/s and a fixed packet rate of 16 p/s. The results of this large simulation are compared to the 500m × 500m and 1500m × 1500m network environments with 50 nodes and 20 connections, a fixed velocity of 11.17 m/s and a fixed packet rate of 16 p/s. The comparison in Figure 32 represents the three different network environments. The smallest network environment of 500m × 500m yielded the least total loss with an average total loss of 3500 packets while the larger network environments of 1500m × 1500m and 2500m × 2500m yielded a total loss of 5000 and 7000 packets, respectively.

Total losses as a function of the offered load for the 500m × 500m network environment is shown in Figure 33. The offered load is represented by the rate at which packets are being sent by the source node. Packet rates of 1, 2, 3, 4, 10, 16 and 20 p/s were used. These packet rates correspond with the following offered load in kilo bits per second (kbps): 81.920, 163.840, 245.760, 327.680, 819.200, 1310.720 and 1638.400, respectively. Small offered loads below 327.680 kbps yielded a total loss of less than 100 packets while offered loads over 1,638.400 kbps yielded a total loss of 5000 packets.

In general, an increase in the size of the network environment increased the amount of total losses. Additionally, an increase in the packet rate contributed to the amount of total losses. An increase in the offered load increased the amount of total losses.
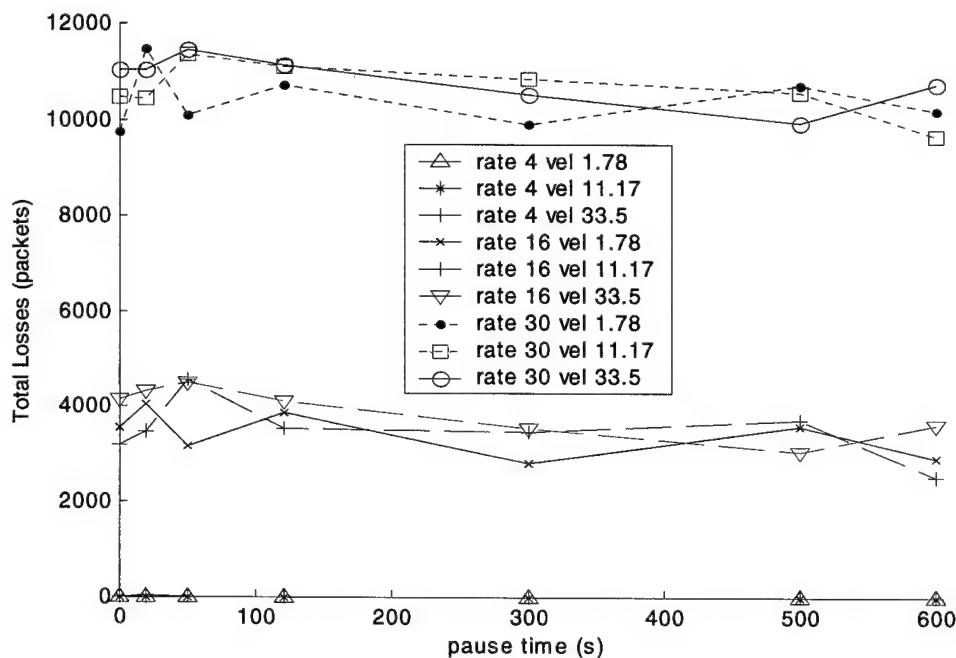
Figure 30. Total Losses for a Network Environment of 500m × 500m with Varied Packet Rates and Node Velocities.
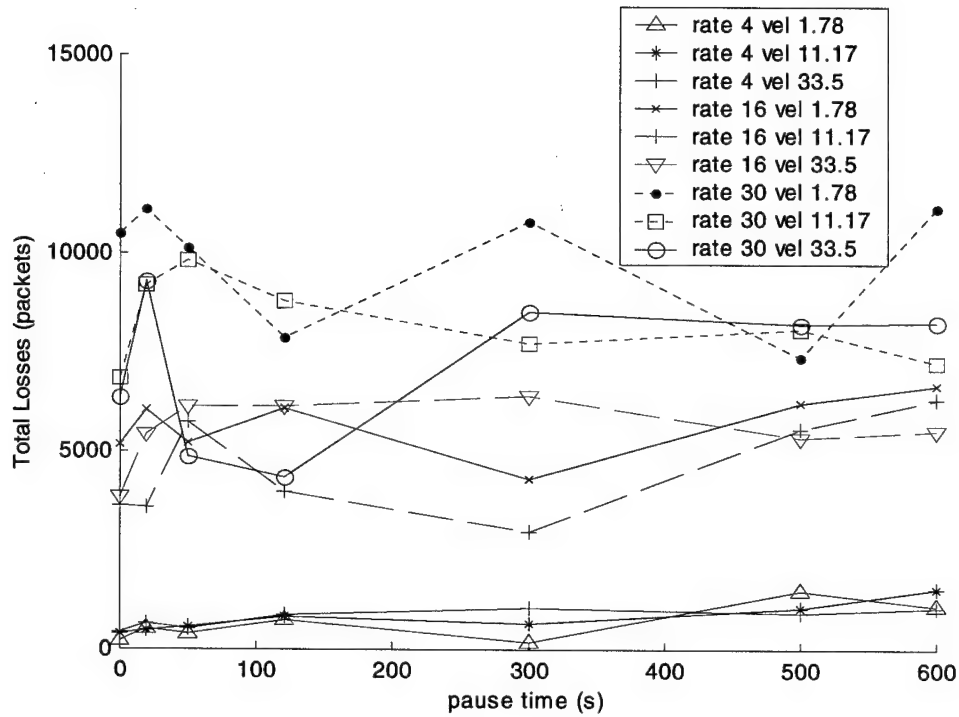
Figure 31. Total Losses for a Network Environment of 1500m × 1500m with Varied Packet Rates and Node Velocities.

Figure 32. Total Losses for Varied Network Environments with a Fixed Packet Rate of 16 p/s and a Fixed Velocity of 11.17 m/s.

46

Figure 33. Total Losses for a Network Environment of 500m × 500m with a Fixed
Pause Time (300 s) and Velocity (11.17 m/s) Having a Varied Packet Rate/Offered
Load.

5. Throughput

The *throughput* is measured as a ratio between the actual number of packets sent
by the source node and the total time taken to transfer these packets. The transfer time is
a sum of the actual time taken to transmit the packets and the overhead time incurred in
implementing message request and flow control mechanisms. Data packets dropped
enroute to the destination are not taken into consideration for this metric.

Figures 34 and 35 show plots of the throughput for a network environment of
500m × 500m and 1500m × 1500m, velocities of 1.78 m/s, 11.17 m/s and 33.5 m/s, and
packet rates of 4, 16, and 30 p/s. All other parameters remained the same as listed in
Table 1. The results of the 500m × 500m network environment (see Figure 34) indicate
that the throughput changes as a function of the packet rates; node velocities did not
affect the throughput. The 1500m × 1500m network environment (see Figure 35)

47

returned results similar to those of the 500m × 500m network environment, but with smaller throughput values.

Throughput for a network environment of 2500m × 2500m with 100 nodes, 30 connections, a fixed velocity of 11.17 m/s and a fixed packet rate of 16 p/s is shown in Figure 36. For comparison, throughput curves of the 500m × 500m and 1500m × 1500m network environments with 50 nodes and 20 connections, respectively, a fixed velocity of 11.17 m/s and a fixed packet rate of 16 p/s are also included. The smallest network environment of 500m × 500m yielded the best throughput with an average of 56 kbps while the larger network environments of 1500m × 1500m and 2500m × 2500m yielded an average throughput of 48 and 55.5 kbps, respectively.

In general, an increase in the size of the network environment decreased the amount of throughput. Larger the network environments increase the difficulty in attaining and maintaining requested destinations causing a decrease in throughput. Additionally, an increase in the packet rate contributed to the amount of throughput.
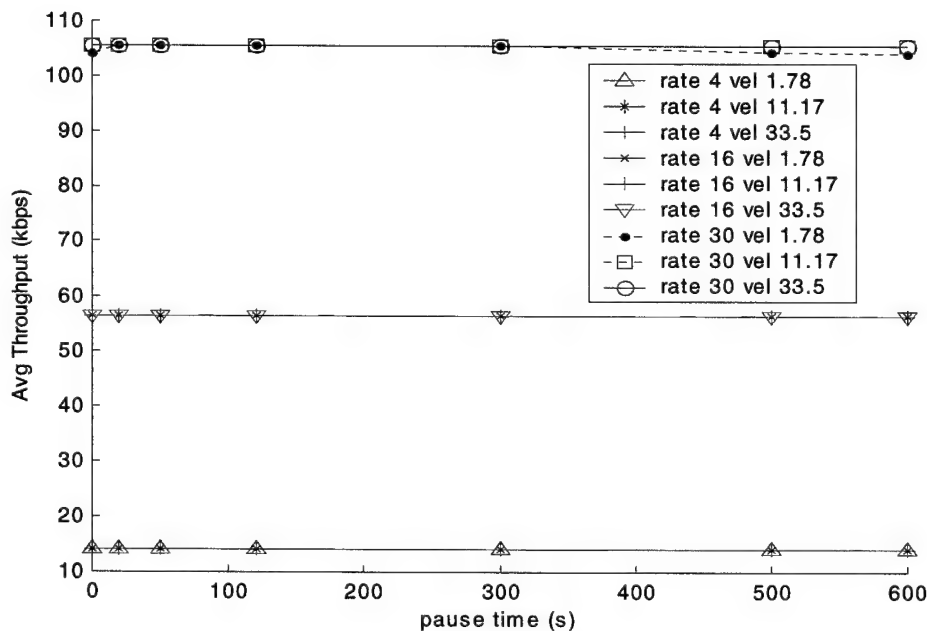
Figure 34. Throughput for a Network Environment of 500m × 500m with Varied Packet Rates and Node Velocities.

48
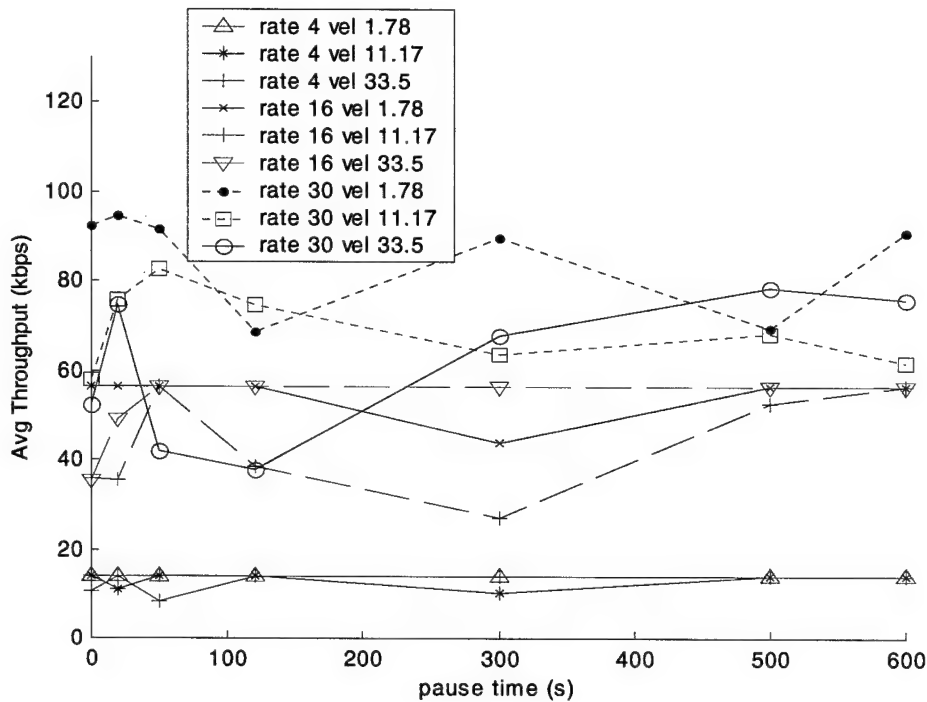
Figure 35. Throughput for a Network Environment of 1500m × 1500m with Varied
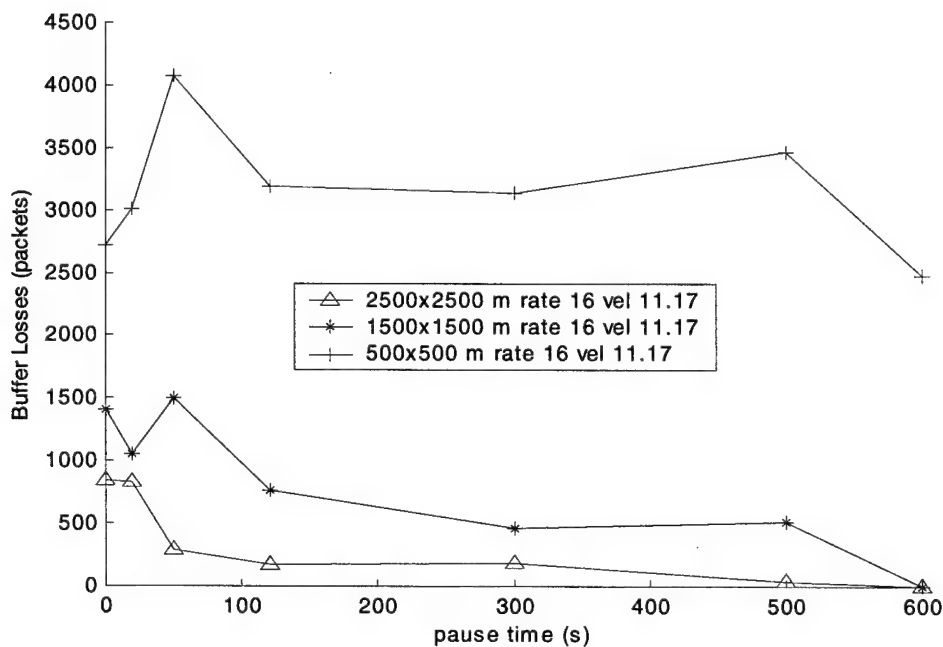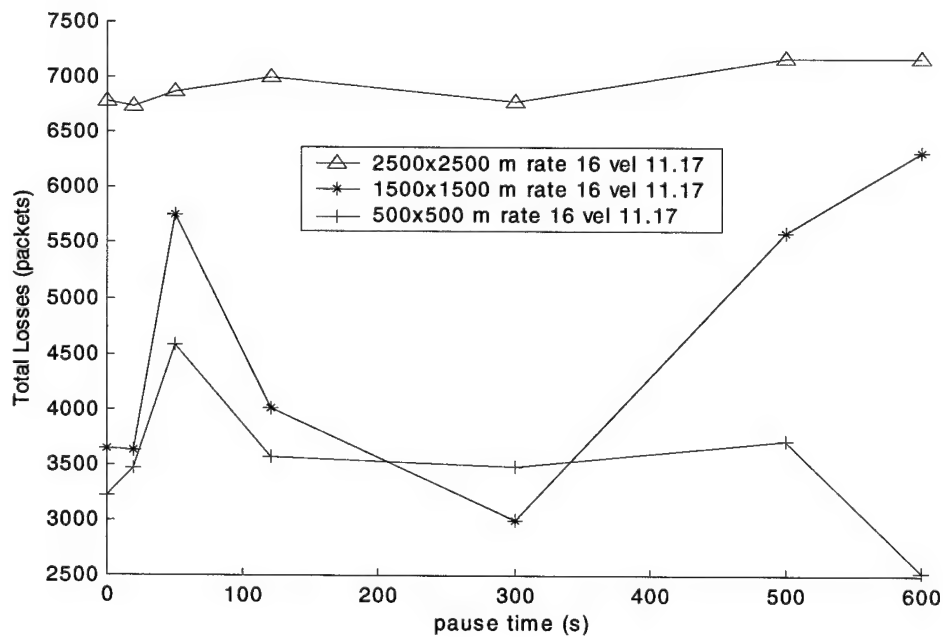Packet Rates and Node Velocities.
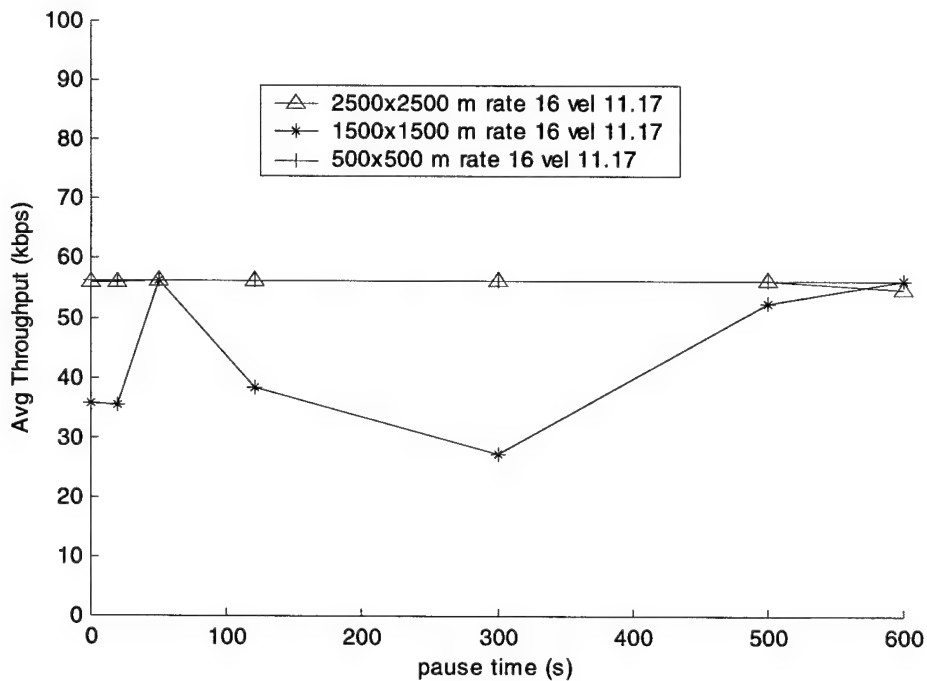
Figure 36. Throughput for Varied Network Environments with a Fixed Packet Rate
of 16 p/s and a Fixed Velocity of 11.17 m/s.

49

6. Goodput

The *goodput* is measured as a ratio between the actual number of packets received by the destination node and the total time to transfer these packets. The transfer time is a sum of the actual time taken to transmit the packets and the overhead time incurred in implementing message request and flow control mechanisms. The total number of packets received by the destination is a true indicator of the performance of this routing protocol and is the best metric to measure its performance.

Figures 37 and 38 present the plots of goodput for a network environment of 500m × 500m and 1500m × 1500m, velocities of 1.78 m/s, 11.17 m/s and 33.5 m/s, and packet rates of 4, 16, and 30 p/s. All other parameters remained the same as listed in Table 1. The results of the 500m × 500m network environment (see Figure 37) indicate that the goodput changes as a function of the packet rates; node velocities did not affect the goodput. Average goodputs of 14, 33 and 33 kbps are obtained for packet rates of 4 p/s, 16 p/s and 30 p/s, respectively. The 1500m × 1500m network environment (see Figure 38) returned results similar to those of the 500m × 500m network environment, but with lower goodput values. Average goodputs of 7, 14, and 15 kbps are obtained for packet rates of 4 p/s, 16 p/s and 30 p/s, respectively.

Goodput for a network environment of 2500m × 2500m with 100 nodes, 30 connections, a fixed velocity of 11.17 m/s and a fixed packet rate of 16 p/s is shown in Figure 39. For comparison, goodput curves of the 500m × 500m and 1500m × 1500m network environments with 50 nodes and 20 connections, respectively, a fixed velocity of 11.17 m/s and a fixed packet rate of 16 p/s are also included. The smallest network environment of 500m × 500m yielded the best goodput with an average of 33 kbps while the larger network environments of 1500m × 1500m and 2500m × 2500m yielded an average goodput of 11 and 8 kbps, respectively.

Figure 40 shows goodput as a function of the offered load for the 500m × 500m network environment. Offered loads below 81 kbps yielded an average goodput of 4 kbps while large offered loads over 1,638 kbps yielded an average goodput of 33 kbps.

50

The increase in the offered load reached a saturation point at 1,310 kbps. Offered loads in excess of 1,638 kbps did not produce a better goodput.

In general, an increase in the size of the network environment decreased the amount of goodput. Additionally, an increase in the packet rate contributed to the amount of goodput. For all network environments, packet rates in excess of 16 p/s yielded approximately the same goodput.
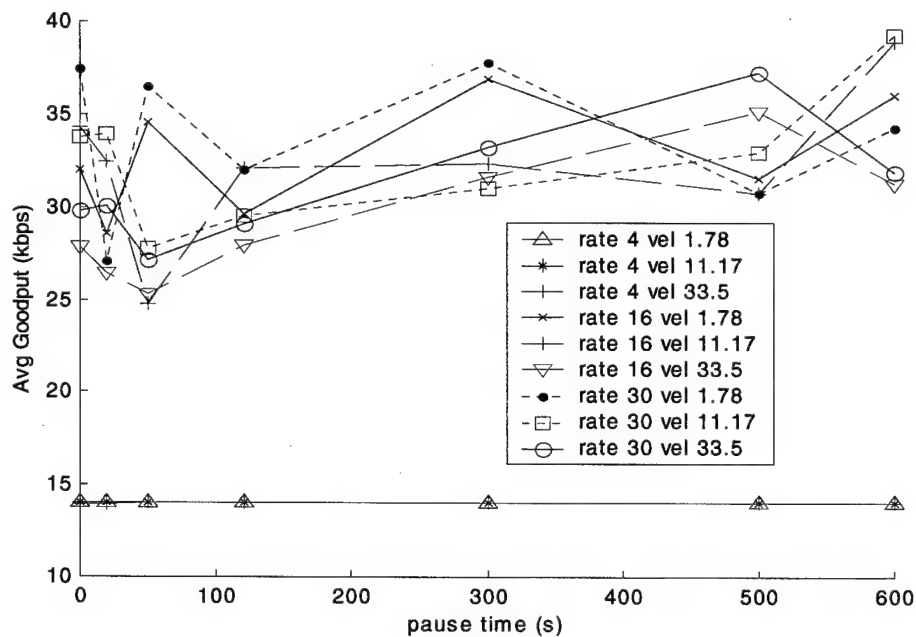
Figure 37. Goodput for a Network Environment of 500m × 500m with Varied Packet Rates and Node Velocities.
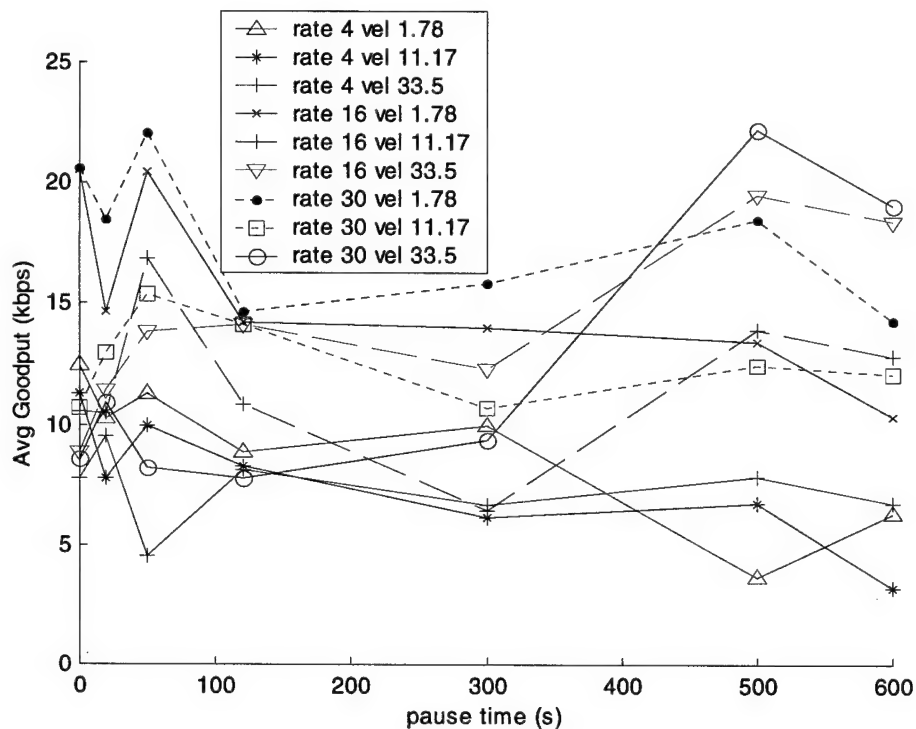
Figure 38. Goodput for a Network Environment of 1500m × 1500m with Varied Packet Rates and Node Velocities.
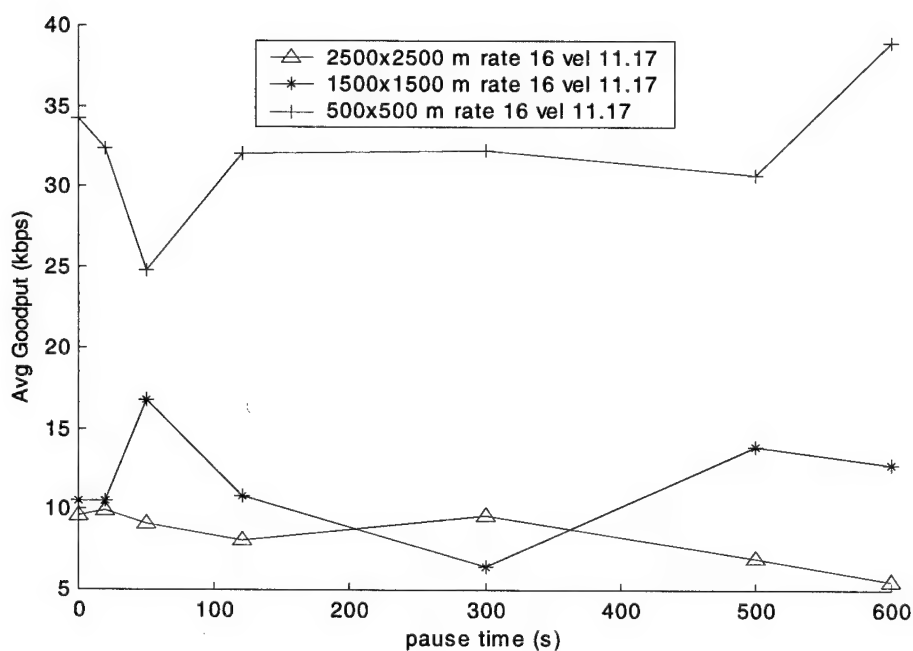
Figure 39. Goodput for Varied Network Environments with a Fixed Packet Rate of 16 p/s and a Fixed Velocity of 11.17 m/s.
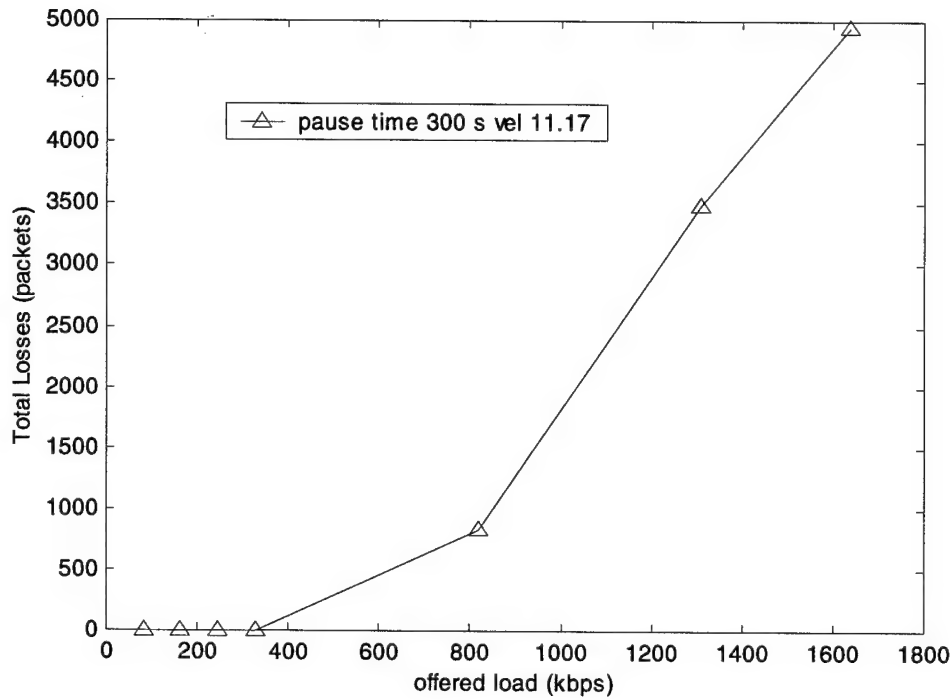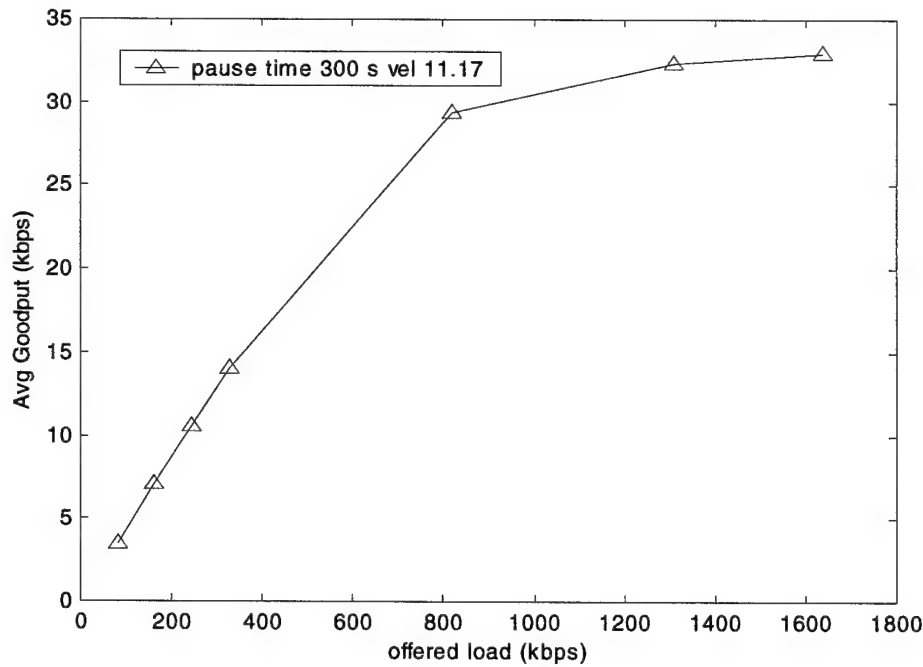
Figure 40. Goodput for a Network Environment of 500m × 500m with a Fixed
Pause Time (300 s) and Velocity (11.17 m/s) Having a Varied Packet Rate/Offered

C.   SUMMARY

Numerous simulations were performed to evaluate the AODV routing protocol.
Two network environments of 500m × 500m and 1500m × 1500m were tested
extensively using various pause times, packet rates, node velocities and offered loads.
All remaining parameters used are listed in Table 1. A small network environment of
500m × 500m provided the best results, but the 1500m × 1500m environment reflects a
more realistic network environment and provides more meaningful results on the
performance of the AODV routing protocol. Additionally, a large network environment
of 2500m × 2500m was simulated for limited cases with a fixed pause time of 300 s and a
fixed velocity of 11.17 m/s. Processing power limitations of the LINUX machine and the
time required to process the results were limiting factors in performing further testing of
network environments in excess of 2500m × 2500m.

Plots of the packet delivery fraction, route loss, buffer loss, total loss, throughput and goodput are presented for the three network environments. Packet rates and the network environment size played significant roles in all these metrics while the node velocity did not have measurable influence. Offered load is an additional parameter that played an important role, but was only evaluated for three metrics: the packet delivery fraction, total losses and goodput.

VI. CONCLUSIONS AND RECOMMENDATIONS

In this thesis, the AODV routing protocol was simulated for three network environments of 500m × 500m, 1500m × 1500m and 2500m × 2500m using varied velocities, packet rates and offered loads to determine its performance. The focus of the performance analysis was on the packet delivery fraction, routing loss, buffer loss, total losses, throughput and goodput.

A.   CONCLUSIONS

The results of this thesis are based on extensive simulation of three network environments in determining the performance of the AODV routing protocol. The 500m × 500m and the 1500m × 1500m network environments were simulated using 50 nodes with a maximum of 20 connections. In the case of 500m × 500m, AODV performed extremely well. All losses were kept at a minimum depending on the packet rate. An increase in the network environment size degraded the overall performance in terms of packet delivery fraction and goodput. However, this degradation in performance was expected for larger network environments because of the difficulty in maintaining routes to required destination nodes. The 2500m × 2500m network environment was simulated using 100 nodes with a maximum of 30 connections. The results produced were preliminary but demonstrated that an increase of the number of nodes and connections did not seriously degrade the overall performance. The results produced were similar to those for the 1500m × 1500m network environment. The results indicate that the most critical parameters are the packet rate and size of the network environment; node velocity and pause time did not affect the protocol performance significantly in most cases. Results produced with the offered load indicate that a large load significantly affects the performance of the packet delivery fraction, total loss and goodput.

55

B.   RECOMMENDATIONS

For future studies of AODV, the following suggestions are offered.   We recommend that the performance of AODV be studied for three network environments using 500m × 500m, 1500m × 1500m and 2500m × 2500m with a range of nodes from 50 to 200.   Also, it is important to consider scenarios in which the number of active connections are close to the number of nodes in the network in order to study the heavy network load conditions.   Finally, power consumption issues and security concerns are to be investigated for these network scenarios.

AODV is a hybrid routing protocol with potential for application in the civilian and the military environments.   Additional routing protocols such as DSR, TORA and ZRP are being considered for use in the JTRS program.   More extensive comparison testing of all these protocols is still required.   This thesis has provided numerous simulation results in evaluating the AODV routing protocol and is recommended as a viable routing protocol for application in a MANET environment as part of the JTRS program.   Nevertheless, additional work is required to determine which MANET routing protocol has the most desirable performance for these environments.

APPENDIX A. TCL SCRIPT FILE

This appendix contains an example of the TCL script used in the simulation. A TCL script file resident in NS2 was modified for the specific purpose of running the parameters listed previously in Table 1.

```
# Copyright (c) 1997 Regents of the University of California.
# All rights reserved.
# Redistribution and use in source and binary forms, with or without
# modification, are permitted provided that the following conditions
# are met:
# 1. Redistributions of source code must retain the above copyright
#    notice, this list of conditions and the following disclaimer.
# 2. Redistributions in binary form must reproduce the above copyright
#    notice, this list of conditions and the following disclaimer in the
#    documentation and/or other materials provided with the distribution.
# 3. All advertising materials mentioning features or use of this software
#    must display the following acknowledgement:
#       This product includes software developed by the Computer Systems
#       Engineering Group at Lawrence Berkeley Laboratory.
# 4. Neither the name of the University nor of the Laboratory may be used
#    to endorse or promote products derived from this software without
#    specific prior written permission.
# THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS
#`AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT
# NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY
# AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO
# EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY
# DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
# CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
# PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
# DATA, OR PROFITS; OR BUSINESS  INTERRUPTION) HOWEVER CAUSED
# AND ON ANY THEORY OF  LIABILITY, WHETHER IN CONTRACT, STRICT
# LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
# IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF
# THE POSSIBILITY OF SUCH DAMAGE.
# $Header: /nfs/jade/vint/CVSROOT/ns-2/tcl/ex/wireless-test.tcl,v 1.5 2000/08/18
# 18:34:04 haoboy Exp $
# Default Script Options
# ========================================================================
set opt(chan)          Channel/WirelessChannel
set opt(prop)          Propagation/TwoRayGround
```

```
set opt(netif)          Phy/WirelessPhy
set opt(mac)            Mac/802_11
set opt(ifq)            Queue/DropTail/PriQueue
set opt(ll)             LL
set opt(ant)            Antenna/OmniAntenna
set opt(x)              500    ;# X dimension of the topography
set opt(y)              500    ;# Y dimension of the topography
set opt(cp)             "../mobility/scene/cbr-50-rate4-ty"
set opt(sc)             "../mobility/scene/scen50-p300-v1.78-move1"
set opt(ifqlen)         64             ;# max packet in ifq
set opt(nn)             50             ;# number of nodes
set opt(seed)           0.0
set opt(stop)           600.0          ;# simulation time
set opt(tr)             out-test.tr           ;# trace file
set opt(rp)             AODV         ;# routing protocol script
set opt(lm)             "off"        ;# log movement
#=================================================================
set AgentTrace              ON
set RouterTrace             ON
set MacTrace                OFF
LL set mindelay_            50us
LL set delay_              25us
LL set bandwidth_           0        ;# not used
Agent/Null set sport_       0
Agent/Null set dport_       0
Agent/CBR set sport_        0
Agent/CBR set dport_        0
Agent/TCPSink set sport_    0
Agent/TCPSink set dport_    0
Agent/TCP set sport_        0
Agent/TCP set dport_        0
Agent/TCP set packetSize_   1460
Queue/DropTail/PriQueue set Prefer_Routing_Protocols    1
# unity gain, omni-directional antennas
# set up the antennas to be centered in the node and 1.5 meters above it
Antenna/OmniAntenna set X_ 0
Antenna/OmniAntenna set Y_ 0
Antenna/OmniAntenna set Z_ 1.5
Antenna/OmniAntenna set Gt_ 1.0
Antenna/OmniAntenna set Gr_ 1.0
# Initialize the SharedMedia interface with parameters to make
# it work like the 914MHz Lucent WaveLAN DSSS radio interface
Phy/WirelessPhy set CPThresh_ 10.0
Phy/WirelessPhy set CSThresh_ 1.559e-11
```

```tcl
Phy/WirelessPhy set RXThresh_ 3.652e-10
Phy/WirelessPhy set Rb_ 2X1e6
Phy/WirelessPhy set Pt_ 0.2818
Phy/WirelessPhy set freq_ 914e+6
Phy/WirelessPhy set L_ 1.0
#=================================================
proc usage { argv0 }  {
        puts "Usage: $argv0"
        puts "\tmandatory arguments:"
        puts "\t\t\[-x MAXX\] \[-y MAXY\]"
        puts "\toptional arguments:"
        puts "\t\t\[-cp conn pattern\] \[-sc scenario\] \[-nn nodes\]"
        puts "\t\t\[-seed seed\] \[-stop sec\] \[-tr tracefile\]\n"
}
proc getopt {argc argv} {
        global opt
        lappend optlist cp nn seed sc stop tr x y
        for {set i 0} {$i < $argc} {incr i} {
                set arg [lindex $argv $i]
                if {[string range $arg 0 0] != "-"} continue
                set name [string range $arg 1 end]
                set opt($name) [lindex $argv [expr $i+1]]
        }
}
proc cmu-trace { ttype atype node } {
        global ns_ tracefd
        if { $tracefd == "" } {
                return ""
        }
        set T [new CMUTrace/$ttype $atype]
        $T target [$ns_ set nullAgent_]
        $T attach $tracefd
    $T set src_ [$node id]
    $T node $node
        return $T
}
proc create-god { nodes } {
        global ns_ god_ tracefd
        set god_ [new God]
        $god_ num_nodes $nodes
}
proc log-movement {} {
   global logtimer ns_ ns
   set ns $ns_
```

```tcl
    source ../mobility/timer.tcl
    Class LogTimer -superclass Timer
    LogTimer instproc timeout {} {
        global opt node_;
        for {set i 0} {$i < $opt(nn)} {incr i} {
            $node_($i) log-movement
        }
        $self sched 0.1
    }
    set logtimer [new LogTimer]
    $logtimer sched 0.1
}
# ===================================================================
# Main Program
# ===================================================================
getopt $argc $argv
getopt $argc $argv
if { $opt(x) == 0 || $opt(y) == 0 } {
        usage $argv0
        exit 1
}
if {$opt(seed) > 0} {
        puts "Seeding Random number generator with $opt(seed)\n"
        ns-random $opt(seed)
}
# Initialize Global Variables
set ns_          [new Simulator]
set chan         [new $opt(chan)]
set prop         [new $opt(prop)]
set topo                    [new Topography]
set tracefd      [open $opt(tr) w]
$topo load_flatgrid $opt(x) $opt(y)
$prop topography $topo
# Create God
create-god $opt(nn)
if { $opt(lm) == "on" } {
   log-movement
}
# Create the specified number of nodes $opt(nn) and "attach" them the channel.  Each
# routing protocol script is expected to have defined a proc create-mobile-node that
# builds a mobile node and inserts it into the array global $node_($i)
$ns_ node-config -adhocRouting $opt(rp) \
                -llType $opt(ll) \
                -macType $opt(mac) \
```

```tcl
                  -ifqType $opt(ifq) \
                  -ifqLen $opt(ifqlen) \
                  -antType $opt(ant) \
                  -propType $opt(prop) \
                  -phyType $opt(netif) \
                  -channelType $opt(chan) \
                  -topoInstance $topo \
                  -agentTrace ON \
                  -routerTrace ON \
                  -macTrace OFF
# Create the specified number of nodes [$opt(nn)] and "attach" them to the channel.
for {set i 0} {$i < $opt(nn) } {incr i} {
        set node_($i) [$ns_ node]
        $node_($i) random-motion 0          ;# disable random motion
}
# Source the Connection and Movement scripts
if { $opt(cp) == "" } {
        puts "XXX NOTE: no connection pattern specified."
    set opt(cp) "none"
} else {
        puts "Loading connection pattern..."
        source $opt(cp)
}
# Tell all the nodes when the simulation ends
for {set i } {$i < $opt(nn) } {incr i} {
   $ns_ at $opt(stop).000000001 "$node_($i) reset";
}
$ns_ at $opt(stop).1 "puts \"NS EXITING...\" ; $ns_ halt"
$ns_ at $opt(stop) "stop"
if { $opt(sc) == "" } {
        puts "XXX NOTE: no scenario file specified."
    set opt(sc) "none"
} else {
        puts "Loading scenario file..."
        source $opt(sc)
        puts "Load complete..."
}
puts $tracefd "M 0.0 nn $opt(nn) x $opt(x) y $opt(y) rp $opt(rp)"
puts $tracefd "M 0.0 sc $opt(sc) cp $opt(cp) seed $opt(seed)"
puts $tracefd "M 0.0 prop $opt(prop) ant $opt(ant)"
puts "Starting Simulation..."
$ns_ run
proc stop {} {
   $ns_ flush-trace
```

61

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B. OUTPUT TRACE FILE

This appendix contains an example of the output trace file generated by the simulator. This output trace file was minized to just a few pages for the reader to see the general output of the trace file. The user must then parse this output using either a grep or awk command, or a perl script to collect the required statistics.

M 0.0 nn 50 x 500 y 500 rp AODV
M 0.0 sc ./mobility/scen50-p50-v11.17-move1 cp ./mobility/cbr-50-rate4-ty seed 0.0
M 0.0 prop Propagation/TwoRayGround ant Antenna/OmniAntenna
s 6.222979895 _7_ AGT --- 0 cbr 512 [0 0 0 0] ------- [7:1 8:1 32 0] [0] 0 1
r 6.222979895 _7_ RTR --- 0 cbr 512 [0 0 0 0] ------- [7:1 8:1 32 0] [0] 0 1
s 6.222979895 _7_ RTR --- 0 AODV 52 [0 0 0 0] ------- [7:255 -1:255 1 0] [0x2 0 1 [8
0] [7 1]] (REQUEST)
r 6.223496010 _5_ RTR --- 0 AODV 52 [0 ffffffff 7 800] ------- [7:255 -1:255 1 0] [0x2
0 1 [8 0] [7 1]] (REQUEST)
D 6.223496010 _5_ RTR TTL 0 AODV 52 [0 ffffffff 7 800] ------- [5:255 -1:255 0 0]
[0x2 1 1 [8 0] [7 1]] (REQUEST)
r 6.223496127 _26_ RTR --- 0 AODV 52 [0 ffffffff 7 800] ------- [7:255 -1:255 1 0] [0x2
0 1 [8 0] [7 1]] (REQUEST)
D 6.223496127 _26_ RTR TTL 0 AODV 52 [0 ffffffff 7 800] ------- [26:255 -1:255 0 0]
[0x2 1 1 [8 0] [7 1]] (REQUEST)
r 6.223496143 _43_ RTR --- 0 AODV 52 [0 ffffffff 7 800] ------- [7:255 -1:255 1 0] [0x2
0 1 [8 0] [7 1]] (REQUEST)
D 6.223496143 _43_ RTR TTL 0 AODV 52 [0 ffffffff 7 800] ------- [43:255 -1:255 0 0]
[0x2 1 1 [8 0] [7 1]] (REQUEST)
r 6.223496157 _18_ RTR --- 0 AODV 52 [0 ffffffff 7 800] ------- [7:255 -1:255 1 0] [0x2
0 1 [8 0] [7 1]] (REQUEST)
D 6.223496157 _18_ RTR TTL 0 AODV 52 [0 ffffffff 7 800] ------- [18:255 -1:255 0 0]
[0x2 1 1 [8 0] [7 1]] (REQUEST)
r 6.223496358 _21_ RTR --- 0 AODV 52 [0 ffffffff 7 800] ------- [7:255 -1:255 1 0] [0x2
0 1 [8 0] [7 1]] (REQUEST)
D 6.223496358 _21_ RTR TTL 0 AODV 52 [0 ffffffff 7 800] ------- [21:255 -1:255 0 0]
[0x2 1 1 [8 0] [7 1]] (REQUEST)
r 6.223496396 _28_ RTR --- 0 AODV 52 [0 ffffffff 7 800] ------- [7:255 -1:255 1 0] [0x2
0 1 [8 0] [7 1]] (REQUEST)
D 6.223496396 _28_ RTR TTL 0 AODV 52 [0 ffffffff 7 800] ------- [28:255 -1:255 0 0]
[0x2 1 1 [8 0] [7 1]] (REQUEST)
r 6.223496410 _33_ RTR --- 0 AODV 52 [0 ffffffff 7 800] ------- [7:255 -1:255 1 0] [0x2
0 1 [8 0] [7 1]] (REQUEST)

D 6.223496410 _33_ RTR TTL 0 AODV 52 [0 ffffffff 7 800] ------- [33:255 -1:255 0 0]
[0x2 1 1 [8 0] [7 1]] (REQUEST)
r 6.223496417 _44_ RTR --- 0 AODV 52 [0 ffffffff 7 800] ------- [7:255 -1:255 1 0] [0x2
0 1 [8 0] [7 1]] (REQUEST)
D 6.223496417 _44_ RTR TTL 0 AODV 52 [0 ffffffff 7 800] ------- [44:255 -1:255 0 0]
[0x2 1 1 [8 0] [7 1]] (REQUEST)
r 6.223496493 _15_ RTR --- 0 AODV 52 [0 ffffffff 7 800] ------- [7:255 -1:255 1 0] [0x2
0 1 [8 0] [7 1]] (REQUEST)
D 6.223496493 _15_ RTR TTL 0 AODV 52 [0 ffffffff 7 800] ------- [15:255 -1:255 0 0]
[0x2 1 1 [8 0] [7 1]] (REQUEST)
r 6.223496495 _8_ RTR --- 0 AODV 52 [0 ffffffff 7 800] ------- [7:255 -1:255 1 0] [0x2
0 1 [8 0] [7 1]] (REQUEST)
s 6.223496495 _8_ RTR --- 0 AODV 44 [0 0 0 0] ------- [8:255 7:255 30 7] [0x4 0 [8 1]
60] (REPLY)
r 6.223496524 _30_ RTR --- 0 AODV 52 [0 ffffffff 7 800] ------- [7:255 -1:255 1 0] [0x2
0 1 [8 0] [7 1]] (REQUEST)
D 6.223496524 _30_ RTR TTL 0 AODV 52 [0 ffffffff 7 800] ------- [30:255 -1:255 0 0]
[0x2 1 1 [8 0] [7 1]] (REQUEST)
r 6.223496580 _16_ RTR --- 0 AODV 52 [0 ffffffff 7 800] ------- [7:255 -1:255 1 0] [0x2
0 1 [8 0] [7 1]] (REQUEST)
D 6.223496580 _16_ RTR TTL 0 AODV 52 [0 ffffffff 7 800] ------- [16:255 -1:255 0 0]
[0x2 1 1 [8 0] [7 1]] (REQUEST)
r 6.223496632 _9_ RTR --- 0 AODV 52 [0 ffffffff 7 800] ------- [7:255 -1:255 1 0] [0x2
0 1 [8 0] [7 1]] (REQUEST)
D 6.223496632 _9_ RTR TTL 0 AODV 52 [0 ffffffff 7 800] ------- [9:255 -1:255 0 0]
[0x2 1 1 [8 0] [7 1]] (REQUEST)
r 6.223496696 _12_ RTR --- 0 AODV 52 [0 ffffffff 7 800] ------- [7:255 -1:255 1 0] [0x2
0 1 [8 0] [7 1]] (REQUEST)
D 6.223496696 _12_ RTR TTL 0 AODV 52 [0 ffffffff 7 800] ------- [12:255 -1:255 0 0]
[0x2 1 1 [8 0] [7 1]] (REQUEST)
r 6.223496720 _23_ RTR --- 0 AODV 52 [0 ffffffff 7 800] ------- [7:255 -1:255 1 0] [0x2
0 1 [8 0] [7 1]] (REQUEST)
D 6.223496720 _23_ RTR TTL 0 AODV 52 [0 ffffffff 7 800] ------- [23:255 -1:255 0 0]
[0x2 1 1 [8 0] [7 1]] (REQUEST)
r 6.226122699 _7_ RTR --- 0 AODV 44 [a2 7 8 800] ------- [8:255 7:255 30 7] [0x4 0 [8
1] 60] (REPLY)
s 6.226122699 _7_ RTR --- 0 cbr 532 [0 0 0 0] ------- [7:1 8:1 30 8] [0] 0 1
r 6.229100500 _8_ AGT --- 0 cbr 532 [a2 8 7 800] ------- [7:1 8:1 30 8] [0] 1 1
s 6.347981852 _7_ AGT --- 1 cbr 512 [0 0 0 0] ------- [7:1 8:1 32 0] [1] 0 1
r 6.347981852 _7_ RTR --- 1 cbr 512 [0 0 0 0] ------- [7:1 8:1 32 0] [1] 0 1
s 6.347981852 _7_ RTR --- 1 cbr 532 [0 0 0 0] ------- [7:1 8:1 30 8] [1] 0 1
r 6.350807653 _8_ AGT --- 1 cbr 532 [a2 8 7 800] ------- [7:1 8:1 30 8] [1] 1 1
s 6.484743006 _7_ AGT --- 2 cbr 512 [0 0 0 0] ------- [7:1 8:1 32 0] [2] 0 1
r 6.484743006 _7_ RTR --- 2 cbr 512 [0 0 0 0] ------- [7:1 8:1 32 0] [2] 0 1

64

s 6.484743006 _7_ RTR --- 2 cbr 532 [0 0 0 0] ------- [7:1 8:1 30 8] [2] 0 1
r 6.487568807 _8_ AGT --- 2 cbr 532 [a2 8 7 800] ------- [7:1 8:1 30 8] [2] 1 1
s 6.779567107 _7_ AGT --- 3 cbr 512 [0 0 0 0] ------- [7:1 8:1 32 0] [3] 0 1
r 6.779567107 _7_ RTR --- 3 cbr 512 [0 0 0 0] ------- [7:1 8:1 32 0] [3] 0 1
s 6.779567107 _7_ RTR --- 3 cbr 532 [0 0 0 0] ------- [7:1 8:1 30 8] [3] 0 1
r 6.782392908 _8_ AGT --- 3 cbr 532 [a2 8 7 800] ------- [7:1 8:1 30 8] [3] 1 1
s 7.000442626 _7_ AGT --- 4 cbr 512 [0 0 0 0] ------- [7:1 8:1 32 0] [4] 0 1
r 7.000442626 _7_ RTR --- 4 cbr 512 [0 0 0 0] ------- [7:1 8:1 32 0] [4] 0 1
s 7.000442626 _7_ RTR --- 4 cbr 532 [0 0 0 0] ------- [7:1 8:1 30 8] [4] 0 1
r 7.003268428 _8_ AGT --- 4 cbr 532 [a2 8 7 800] ------- [7:1 8:1 30 8] [4] 1 1
s 7.333183963 _7_ AGT --- 5 cbr 512 [0 0 0 0] ------- [7:1 8:1 32 0] [5] 0 1
r 7.333183963 _7_ RTR --- 5 cbr 512 [0 0 0 0] ------- [7:1 8:1 32 0] [5] 0 1
s 7.333183963 _7_ RTR --- 5 cbr 532 [0 0 0 0] ------- [7:1 8:1 30 8] [5] 0 1
r 7.336009764 _8_ AGT --- 5 cbr 532 [a2 8 7 800] ------- [7:1 8:1 30 8] [5] 1 1
s 7.471549372 _7_ AGT --- 6 cbr 512 [0 0 0 0] ------- [7:1 8:1 32 0] [6] 0 1
r 7.471549372 _7_ RTR --- 6 cbr 512 [0 0 0 0] ------- [7:1 8:1 32 0] [6] 0 1
s 7.471549372 _7_ RTR --- 6 cbr 532 [0 0 0 0] ------- [7:1 8:1 30 8] [6] 0 1
r 7.474375173 _8_ AGT --- 6 cbr 532 [a2 8 7 800] ------- [7:1 8:1 30 8] [6] 1 1
s 7.764336718 _7_ AGT --- 7 cbr 512 [0 0 0 0] ------- [7:1 8:1 32 0] [7] 0 1
r 7.764336718 _7_ RTR --- 7 cbr 512 [0 0 0 0] ------- [7:1 8:1 32 0] [7] 0 1
s 7.764336718 _7_ RTR --- 7 cbr 532 [0 0 0 0] ------- [7:1 8:1 30 8] [7] 0 1
r 7.767162519 _8_ AGT --- 7 cbr 532 [a2 8 7 800] ------- [7:1 8:1 30 8] [7] 1 1
s 7.985190630 _7_ AGT --- 8 cbr 512 [0 0 0 0] ------- [7:1 8:1 32 0] [8] 0 1
r 7.985190630 _7_ RTR --- 8 cbr 512 [0 0 0 0] ------- [7:1 8:1 32 0] [8] 0 1
s 7.985190630 _7_ RTR --- 8 cbr 532 [0 0 0 0] ------- [7:1 8:1 30 8] [8] 0 1
r 7.988016432 _8_ AGT --- 8 cbr 532 [a2 8 7 800] ------- [7:1 8:1 30 8] [8] 1 1
s 8.214562124 _7_ AGT --- 9 cbr 512 [0 0 0 0] ------- [7:1 8:1 32 0] [9] 0 1
r 8.214562124 _7_ RTR --- 9 cbr 512 [0 0 0 0] ------- [7:1 8:1 32 0] [9] 0 1
s 8.214562124 _7_ RTR --- 9 cbr 532 [0 0 0 0] ------- [7:1 8:1 30 8] [9] 0 1
r 8.217387925 _8_ AGT --- 9 cbr 532 [a2 8 7 800] ------- [7:1 8:1 30 8] [9] 1 1
s 8.486806285 _7_ AGT --- 10 cbr 512 [0 0 0 0] ------- [7:1 8:1 32 0] [10] 0 1
r 8.486806285 _7_ RTR --- 10 cbr 512 [0 0 0 0] ------- [7:1 8:1 32 0] [10] 0 1
s 8.486806285 _7_ RTR --- 10 cbr 532 [0 0 0 0] ------- [7:1 8:1 30 8] [10] 0 1
r 8.489632086 _8_ AGT --- 10 cbr 532 [a2 8 7 800] ------- [7:1 8:1 30 8] [10] 1 1
s 8.543612467 _18_ AGT --- 11 cbr 512 [0 0 0 0] ------- [18:2 20:0 32 0] [0] 0 2
r 8.543612467 _18_ RTR --- 11 cbr 512 [0 0 0 0] ------- [18:2 20:0 32 0] [0] 0 2
s 8.543612467 _18_ RTR --- 0 AODV 52 [0 0 0 0] ------- [18:255 -1:255 1 0] [0x2 0 1 [20 0] [18 1]] (REQUEST)
r 8.544128730 _7_ RTR --- 0 AODV 52 [0 ffffffff 12 800] ------- [18:255 -1:255 1 0] [0x2 0 1 [20 0] [18 1]] (REQUEST)
D 8.544128730 _7_ RTR TTL 0 AODV 52 [0 ffffffff 12 800] ------- [7:255 -1:255 0 0] [0x2 1 1 [20 0] [18 1]] (REQUEST)
r 8.544128771 _43_ RTR --- 0 AODV 52 [0 ffffffff 12 800] ------- [18:255 -1:255 1 0] [0x2 0 1 [20 0] [18 1]] (REQUEST)

65

D 8.544128771 _43_ RTR TTL 0 AODV 52 [0 ffffffff 12 800] ------- [43:255 -1:255 0
0] [0x2 1 1 [20 0] [18 1]] (REQUEST)
r 8.544128805 _15_ RTR --- 0 AODV 52 [0 ffffffff 12 800] ------- [18:255 -1:255 1 0]
[0x2 0 1 [20 0] [18 1]] (REQUEST)
D 8.544128805 _15_ RTR TTL 0 AODV 52 [0 ffffffff 12 800] ------- [15:255 -1:255 0
0] [0x2 1 1 [20 0] [18 1]] (REQUEST)
r 8.544128816 _5_ RTR --- 0 AODV 52 [0 ffffffff 12 800] ------- [18:255 -1:255 1 0]
[0x2 0 1 [20 0] [18 1]] (REQUEST)
D 8.544128816 _5_ RTR TTL 0 AODV 52 [0 ffffffff 12 800] ------- [5:255 -1:255 0 0]
[0x2 1 1 [20 0] [18 1]] (REQUEST)
r 8.544128895 _16_ RTR --- 0 AODV 52 [0 ffffffff 12 800] ------- [18:255 -1:255 1 0]
[0x2 0 1 [20 0] [18 1]] (REQUEST)
D 8.544128895 _16_ RTR TTL 0 AODV 52 [0 ffffffff 12 800] ------- [16:255 -1:255 0
0] [0x2 1 1 [20 0] [18 1]] (REQUEST)
r 8.544128902 _26_ RTR --- 0 AODV 52 [0 ffffffff 12 800] ------- [18:255 -1:255 1 0]
[0x2 0 1 [20 0] [18 1]] (REQUEST)
D 8.544128902 _26_ RTR TTL 0 AODV 52 [0 ffffffff 12 800] ------- [26:255 -1:255 0
0] [0x2 1 1 [20 0] [18 1]] (REQUEST)
r 8.544128925 _30_ RTR --- 0 AODV 52 [0 ffffffff 12 800] ------- [18:255 -1:255 1 0]
[0x2 0 1 [20 0] [18 1]] (REQUEST)
D 8.544128925 _30_ RTR TTL 0 AODV 52 [0 ffffffff 12 800] ------- [30:255 -1:255 0
0] [0x2 1 1 [20 0] [18 1]] (REQUEST)
r 8.544128930 _8_ RTR --- 0 AODV 52 [0 ffffffff 12 800] ------- [18:255 -1:255 1 0]
[0x2 0 1 [20 0] [18 1]] (REQUEST)
D 8.544128930 _8_ RTR TTL 0 AODV 52 [0 ffffffff 12 800] ------- [8:255 -1:255 0 0]
[0x2 1 1 [20 0] [18 1]] (REQUEST)
r 8.544128974 _33_ RTR --- 0 AODV 52 [0 ffffffff 12 800] ------- [18:255 -1:255 1 0]
[0x2 0 1 [20 0] [18 1]] (REQUEST)
D 8.544128974 _33_ RTR TTL 0 AODV 52 [0 ffffffff 12 800] ------- [33:255 -1:255 0
0] [0x2 1 1 [20 0] [18 1]] (REQUEST)
r 8.544128978 _28_ RTR --- 0 AODV 52 [0 ffffffff 12 800] ------- [18:255 -1:255 1 0]
[0x2 0 1 [20 0] [18 1]] (REQUEST)
D 8.544128978 _28_ RTR TTL 0 AODV 52 [0 ffffffff 12 800] ------- [28:255 -1:255 0
0] [0x2 1 1 [20 0] [18 1]] (REQUEST)
r 8.544129010 _12_ RTR --- 0 AODV 52 [0 ffffffff 12 800] ------- [18:255 -1:255 1 0]
[0x2 0 1 [20 0] [18 1]] (REQUEST)
D 8.544129010 _12_ RTR TTL 0 AODV 52 [0 ffffffff 12 800] ------- [12:255 -1:255 0
0] [0x2 1 1 [20 0] [18 1]] (REQUEST)
r 8.544129054 _9_ RTR --- 0 AODV 52 [0 ffffffff 12 800] ------- [18:255 -1:255 1 0]
[0x2 0 1 [20 0] [18 1]] (REQUEST)
D 8.544129054 _9_ RTR TTL 0 AODV 52 [0 ffffffff 12 800] ------- [9:255 -1:255 0 0]
[0x2 1 1 [20 0] [18 1]] (REQUEST)
r 8.544129091 _4_ RTR --- 0 AODV 52 [0 ffffffff 12 800] ------- [18:255 -1:255 1 0]
[0x2 0 1 [20 0] [18 1]] (REQUEST)

D 8.544129091 _4_ RTR TTL 0 AODV 52 [0 ffffffff 12 800] ------- [4:255 -1:255 0 0] [0x2 1 1 [20 0] [18 1]] (REQUEST)

r 8.544129094 _27_ RTR --- 0 AODV 52 [0 ffffffff 12 800] ------- [18:255 -1:255 1 0] [0x2 0 1 [20 0] [18 1]] (REQUEST)

D 8.544129094 _27_ RTR TTL 0 AODV 52 [0 ffffffff 12 800] ------- [27:255 -1:255 0 0] [0x2 1 1 [20 0] [18 1]] (REQUEST)

r 8.544129104 _37_ RTR --- 0 AODV 52 [0 ffffffff 12 800] ------- [18:255 -1:255 1 0] [0x2 0 1 [20 0] [18 1]] (REQUEST)

D 8.544129104 _37_ RTR TTL 0 AODV 52 [0 ffffffff 12 800] ------- [37:255 -1:255 0 0] [0x2 1 1 [20 0] [18 1]] (REQUEST)

r 8.544129112 _11_ RTR --- 0 AODV 52 [0 ffffffff 12 800] ------- [18:255 -1:255 1 0] [0x2 0 1 [20 0] [18 1]] (REQUEST)

D 8.544129112 _11_ RTR TTL 0 AODV 52 [0 ffffffff 12 800] ------- [11:255 -1:255 0 0] [0x2 1 1 [20 0] [18 1]] (REQUEST)

r 8.544129117 _6_ RTR --- 0 AODV 52 [0 ffffffff 12 800] ------- [18:255 -1:255 1 0] [0x2 0 1 [20 0] [18 1]] (REQUEST)

D 8.544129117 _6_ RTR TTL 0 AODV 52 [0 ffffffff 12 800] ------- [6:255 -1:255 0 0] [0x2 1 1 [20 0] [18 1]] (REQUEST)

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C. NODE MOVEMENT FILE

This appendix contains an example of the node movement file used in the simulation that was generated by the user. A command line input into NS2 generated the node movement file based upon specific parameters established by the user. The desription of the node movement file is contained in chapter 4. The shaded region depicts a sample command line input used in NS2.

```
./setdest –n 50 –p 500 –s 1.78 –t 600 –x 1500 –y 300 > scen50-vel1.78-ty
```

```
#
# nodes: 50, pause: 700.00, max speed: 1.78  max x = 500.00, max y: 500.00
#
$node_(0) set X_ 298.272428225941
$node_(0) set Y_ 64.701458000283
$node_(0) set Z_ 0.000000000000
$node_(1) set X_ 437.404668139211
$node_(1) set Y_ 460.257803745608
$node_(1) set Z_ 0.000000000000
$node_(2) set X_ 52.907960744067
$node_(2) set Y_ 224.096272466096
$node_(2) set Z_ 0.000000000000
$node_(3) set X_ 386.051536464268
$node_(3) set Y_ 368.173697416875
$node_(3) set Z_ 0.000000000000
$node_(4) set X_ 395.332812029179
$node_(4) set Y_ 358.572125117260
$node_(4) set Z_ 0.000000000000
$node_(5) set X_ 21.707163899786
$node_(5) set Y_ 332.303682945033
$node_(5) set Z_ 0.000000000000
$node_(6) set X_ 27.999551978347
$node_(6) set Y_ 88.470124908929
$node_(6) set Z_ 0.000000000000
$node_(7) set X_ 417.389422827436
$node_(7) set Y_ 64.029831000032
$node_(7) set Z_ 0.000000000000
$node_(8) set X_ 149.369674330781
$node_(8) set Y_ 456.116609929322
$node_(8) set Z_ 0.000000000000
```

69

$node_(9) set X_ 451.863486739175
$node_(9) set Y_ 469.622026158678
$node_(9) set Z_ 0.000000000000
$node_(10) set X_ 437.394065497356
$node_(10) set Y_ 282.059202087152
$node_(10) set Z_ 0.000000000000
$node_(11) set X_ 69.009728991048
$node_(11) set Y_ 346.515213748085
$node_(11) set Z_ 0.000000000000
$node_(12) set X_ 381.197771447705
$node_(12) set Y_ 290.945059739968
$node_(12) set Z_ 0.000000000000
$node_(13) set X_ 413.619307739071
$node_(13) set Y_ 199.705537491044
$node_(13) set Z_ 0.000000000000
$node_(14) set X_ 450.968789122929
$node_(14) set Y_ 432.439189117132
$node_(14) set Z_ 0.000000000000
$node_(15) set X_ 5.451875275389
$node_(15) set Y_ 129.667758296695
$node_(15) set Z_ 0.000000000000
$node_(16) set X_ 326.013807574780
$node_(16) set Y_ 314.064198536083
$node_(16) set Z_ 0.000000000000
$node_(17) set X_ 476.985074543821
$node_(17) set Y_ 188.148281149017
$node_(17) set Z_ 0.000000000000
$node_(18) set X_ 208.161438435579
$node_(18) set Y_ 69.295971450136
$node_(18) set Z_ 0.000000000000
$node_(19) set X_ 157.392223896250
$node_(19) set Y_ 291.107163884728
$node_(19) set Z_ 0.000000000000
$node_(20) set X_ 138.103668867820
$node_(20) set Y_ 108.362783955447
$node_(20) set Z_ 0.000000000000
$node_(21) set X_ 253.310035320277
$node_(21) set Y_ 381.763852591854
$node_(21) set Z_ 0.000000000000
$node_(22) set X_ 305.070849959967
$node_(22) set Y_ 325.775547786079
$node_(22) set Z_ 0.000000000000
$node_(23) set X_ 309.631929627868
$node_(23) set Y_ 483.841530247124

$node_(23) set Z_ 0.000000000000
$node_(24) set X_ 424.599292630548
$node_(24) set Y_ 240.311618295066
$node_(24) set Z_ 0.000000000000
$node_(25) set X_ 417.368898340528
$node_(25) set Y_ 219.074779513405
$node_(25) set Z_ 0.000000000000
$node_(26) set X_ 489.819476117518
$node_(26) set Y_ 395.935541647978
$node_(26) set Z_ 0.000000000000
$node_(27) set X_ 488.648828786453
$node_(27) set Y_ 220.865847401242
$node_(27) set Z_ 0.000000000000
$node_(28) set X_ 92.297468605375
$node_(28) set Y_ 243.554932398697
$node_(28) set Z_ 0.000000000000
$node_(29) set X_ 427.749040946021
$node_(29) set Y_ 178.131559238741
$node_(29) set Z_ 0.000000000000
$node_(30) set X_ 357.116283530665
$node_(30) set Y_ 53.377616706922
$node_(30) set Z_ 0.000000000000
$node_(31) set X_ 117.604040589311
$node_(31) set Y_ 71.110288874735
$node_(31) set Z_ 0.000000000000
$node_(32) set X_ 150.625180748082
$node_(32) set Y_ 57.412966643882
$node_(32) set Z_ 0.000000000000
$node_(33) set X_ 439.730434627219
$node_(33) set Y_ 49.415169769811
$node_(33) set Z_ 0.000000000000
$node_(34) set X_ 20.758365056666
$node_(34) set Y_ 385.841525784743
$node_(34) set Z_ 0.000000000000
$node_(35) set X_ 338.524206448417
$node_(35) set Y_ 76.338078858184
$node_(35) set Z_ 0.000000000000
$node_(36) set X_ 14.091437222666
$node_(36) set Y_ 334.785413833724
$node_(36) set Z_ 0.000000000000
$node_(37) set X_ 238.450600398435
$node_(37) set Y_ 139.241108029828
$node_(37) set Z_ 0.000000000000
$node_(38) set X_ 225.302780838573

```
$node_(38) set Y_ 163.837753769600
$node_(38) set Z_ 0.000000000000
$node_(39) set X_ 121.127751007396
$node_(39) set Y_ 294.111288739728
$node_(39) set Z_ 0.000000000000
$node_(40) set X_ 128.430109524109
$node_(40) set Y_ 24.850885626876
$node_(40) set Z_ 0.000000000000
$node_(41) set X_ 168.834752938569
$node_(41) set Y_ 105.692788303696
$node_(41) set Z_ 0.000000000000
$node_(42) set X_ 378.693113958355
$node_(42) set Y_ 195.166634010637
$node_(42) set Z_ 0.000000000000
$node_(43) set X_ 165.617989910228
$node_(43) set Y_ 41.556568134970
$node_(43) set Z_ 0.000000000000
$node_(44) set X_ 441.240681284677
$node_(44) set Y_ 432.130742996534
$node_(44) set Z_ 0.000000000000
$node_(45) set X_ 321.397926092598
$node_(45) set Y_ 234.944123406109
$node_(45) set Z_ 0.000000000000
$node_(46) set X_ 205.882294886999
$node_(46) set Y_ 263.730348429566
$node_(46) set Z_ 0.000000000000
$node_(47) set X_ 15.966289683318
$node_(47) set Y_ 345.430721671449
$node_(47) set Z_ 0.000000000000
$node_(48) set X_ 154.139438474862
$node_(48) set Y_ 121.542583735550
$node_(48) set Z_ 0.000000000000
$node_(49) set X_ 266.204951208099
$node_(49) set Y_ 106.615190671072
$node_(49) set Z_ 0.000000000000
```

APPENDIX D. TRAFFIC PATTERN FILE

This appendix contains an example of the traffic pattern file used in the simulation that was generated by the user. A command line input into NS2 generated the traffic pattern file based upon specific parameters established by the user. The desription of the node movement file is contained in chapter 4. The shaded region depicts a sample command line input used in NS2.

ns cbrgen.tcl –type cbr –nn 50 –seed 1.0 –mc 20 –rate 4.0 > cbr-50-rate4-ty

```
# nodes: 50, max conn: 20, send rate: 0.25, seed: 1.0
# 2 connecting to 3 at time 82.557023746220864
set udp_(0) [new Agent/UDP]
$ns_ attach-agent $node_(2) $udp_(0)
set null_(0) [new Agent/Null]
$ns_ attach-agent $node_(3) $null_(0)
set cbr_(0) [new Application/Traffic/CBR]
$cbr_(0) set packetSize_ 512
$cbr_(0) set interval_ 0.25
$cbr_(0) set random_ 1
$cbr_(0) set maxpkts_ 10000
$cbr_(0) attach-agent $udp_(0)
$ns_ connect $udp_(0) $null_(0)
$ns_ at 82.557023746220864 "$cbr_(0) start"
#
# 2 connecting to 4 at time 95.898102734190459
set udp_(1) [new Agent/UDP]
$ns_ attach-agent $node_(2) $udp_(1)
set null_(1) [new Agent/Null]
$ns_ attach-agent $node_(4) $null_(1)
set cbr_(1) [new Application/Traffic/CBR]
$cbr_(1) set packetSize_ 512
$cbr_(1) set interval_ 0.25
$cbr_(1) set random_ 1
$cbr_(1) set maxpkts_ 10000
$cbr_(1) attach-agent $udp_(1)
$ns_ connect $udp_(1) $null_(1)
$ns_ at 95.898102734190459 "$cbr_(1) start"
#
# 5 connecting to 6 at time 122.2733530505902
```

```
set udp_(2) [new Agent/UDP]
$ns_ attach-agent $node_(5) $udp_(2)
set null_(2) [new Agent/Null]
$ns_ attach-agent $node_(6) $null_(2)
set cbr_(2) [new Application/Traffic/CBR]
$cbr_(2) set packetSize_ 512
$cbr_(2) set interval_ 0.25
$cbr_(2) set random_ 1
$cbr_(2) set maxpkts_ 10000
$cbr_(2) attach-agent $udp_(2)
$ns_ connect $udp_(2) $null_(2)
$ns_ at 122.2733530505902 "$cbr_(2) start"
#
# 6 connecting to 7 at time 69.030373948174713
set udp_(3) [new Agent/UDP]
$ns_ attach-agent $node_(6) $udp_(3)
set null_(3) [new Agent/Null]
$ns_ attach-agent $node_(7) $null_(3)
set cbr_(3) [new Application/Traffic/CBR]
$cbr_(3) set packetSize_ 512
$cbr_(3) set interval_ 0.25
$cbr_(3) set random_ 1
$cbr_(3) set maxpkts_ 10000
$cbr_(3) attach-agent $udp_(3)
$ns_ connect $udp_(3) $null_(3)
$ns_ at 69.030373948174713 "$cbr_(3) start"
#
# 6 connecting to 8 at time 93.494946972231816
set udp_(4) [new Agent/UDP]
$ns_ attach-agent $node_(6) $udp_(4)
set null_(4) [new Agent/Null]
$ns_ attach-agent $node_(8) $null_(4)
set cbr_(4) [new Application/Traffic/CBR]
$cbr_(4) set packetSize_ 512
$cbr_(4) set interval_ 0.25
$cbr_(4) set random_ 1
$cbr_(4) set maxpkts_ 10000
$cbr_(4) attach-agent $udp_(4)
$ns_ connect $udp_(4) $null_(4)
$ns_ at 93.494946972231816 "$cbr_(4) start"
#
# 7 connecting to 8 at time 6.2229798949430606
set udp_(5) [new Agent/UDP]
$ns_ attach-agent $node_(7) $udp_(5)
```

```
set null_(5) [new Agent/Null]
$ns_ attach-agent $node_(8) $null_(5)
set cbr_(5) [new Application/Traffic/CBR]
$cbr_(5) set packetSize_ 512
$cbr_(5) set interval_ 0.25
$cbr_(5) set random_ 1
$cbr_(5) set maxpkts_ 10000
$cbr_(5) attach-agent $udp_(5)
$ns_ connect $udp_(5) $null_(5)
$ns_ at 6.2229798949430606 "$cbr_(5) start"
#
# 7 connecting to 9 at time 9.6230943080145384
set udp_(6) [new Agent/UDP]
$ns_ attach-agent $node_(7) $udp_(6)
set null_(6) [new Agent/Null]
$ns_ attach-agent $node_(9) $null_(6)
set cbr_(6) [new Application/Traffic/CBR]
$cbr_(6) set packetSize_ 512
$cbr_(6) set interval_ 0.25
$cbr_(6) set random_ 1
$cbr_(6) set maxpkts_ 10000
$cbr_(6) attach-agent $udp_(6)
$ns_ connect $udp_(6) $null_(6)
$ns_ at 9.6230943080145384 "$cbr_(6) start"
#
# 8 connecting to 9 at time 120.80688913390362
set udp_(7) [new Agent/UDP]
$ns_ attach-agent $node_(8) $udp_(7)
set null_(7) [new Agent/Null]
$ns_ attach-agent $node_(9) $null_(7)
set cbr_(7) [new Application/Traffic/CBR]
$cbr_(7) set packetSize_ 512
$cbr_(7) set interval_ 0.25
$cbr_(7) set random_ 1
$cbr_(7) set maxpkts_ 10000
$cbr_(7) attach-agent $udp_(7)
$ns_ connect $udp_(7) $null_(7)
$ns_ at 120.80688913390362 "$cbr_(7) start"
#
# 13 connecting to 14 at time 106.01579571422924
set udp_(8) [new Agent/UDP]
$ns_ attach-agent $node_(13) $udp_(8)
set null_(8) [new Agent/Null]
$ns_ attach-agent $node_(14) $null_(8)
```

```
set cbr_(8) [new Application/Traffic/CBR]
$cbr_(8) set packetSize_ 512
$cbr_(8) set interval_ 0.25
$cbr_(8) set random_ 1
$cbr_(8) set maxpkts_ 10000
$cbr_(8) attach-agent $udp_(8)
$ns_ connect $udp_(8) $null_(8)
$ns_ at 106.01579571422924 "$cbr_(8) start"
#
# 14 connecting to 15 at time 152.31004029154315
set udp_(9) [new Agent/UDP]
$ns_ attach-agent $node_(14) $udp_(9)
set null_(9) [new Agent/Null]
$ns_ attach-agent $node_(15) $null_(9)
set cbr_(9) [new Application/Traffic/CBR]
$cbr_(9) set packetSize_ 512
$cbr_(9) set interval_ 0.25
$cbr_(9) set random_ 1
$cbr_(9) set maxpkts_ 10000
$cbr_(9) attach-agent $udp_(9)
$ns_ connect $udp_(9) $null_(9)
$ns_ at 152.31004029154315 "$cbr_(9) start"
#
# 14 connecting to 16 at time 94.847179965510577
set udp_(10) [new Agent/UDP]
$ns_ attach-agent $node_(14) $udp_(10)
set null_(10) [new Agent/Null]
$ns_ attach-agent $node_(16) $null_(10)
set cbr_(10) [new Application/Traffic/CBR]
$cbr_(10) set packetSize_ 512
$cbr_(10) set interval_ 0.25
$cbr_(10) set random_ 1
$cbr_(10) set maxpkts_ 10000
$cbr_(10) attach-agent $udp_(10)
$ns_ connect $udp_(10) $null_(10)
$ns_ at 94.847179965510577 "$cbr_(10) start"
#
# 16 connecting to 17 at time 74.879884233176654
set udp_(11) [new Agent/UDP]
$ns_ attach-agent $node_(16) $udp_(11)
set null_(11) [new Agent/Null]
$ns_ attach-agent $node_(17) $null_(11)
set cbr_(11) [new Application/Traffic/CBR]
$cbr_(11) set packetSize_ 512
```

```
$cbr_(11) set interval_ 0.25
$cbr_(11) set random_ 1
$cbr_(11) set maxpkts_ 10000
$cbr_(11) attach-agent $udp_(11)
$ns_ connect $udp_(11) $null_(11)
$ns_ at 74.879884233176654 "$cbr_(11) start"
#
# 17 connecting to 18 at time 163.85774948813847
set udp_(12) [new Agent/UDP]
$ns_ attach-agent $node_(17) $udp_(12)
set null_(12) [new Agent/Null]
$ns_ attach-agent $node_(18) $null_(12)
set cbr_(12) [new Application/Traffic/CBR]
$cbr_(12) set packetSize_ 512
$cbr_(12) set interval_ 0.25
$cbr_(12) set random_ 1
$cbr_(12) set maxpkts_ 10000
$cbr_(12) attach-agent $udp_(12)
$ns_ connect $udp_(12) $null_(12)
$ns_ at 163.85774948813847 "$cbr_(12) start"
#
# 18 connecting to 19 at time 47.241538859550623
set udp_(13) [new Agent/UDP]
$ns_ attach-agent $node_(18) $udp_(13)
set null_(13) [new Agent/Null]
$ns_ attach-agent $node_(19) $null_(13)
set cbr_(13) [new Application/Traffic/CBR]
$cbr_(13) set packetSize_ 512
$cbr_(13) set interval_ 0.25
$cbr_(13) set random_ 1
$cbr_(13) set maxpkts_ 10000
$cbr_(13) attach-agent $udp_(13)
$ns_ connect $udp_(13) $null_(13)
$ns_ at 47.241538859550623 "$cbr_(13) start"
#
# 18 connecting to 20 at time 8.54361246737781979
set udp_(14) [new Agent/UDP]
$ns_ attach-agent $node_(18) $udp_(14)
set null_(14) [new Agent/Null]
$ns_ attach-agent $node_(20) $null_(14)
set cbr_(14) [new Application/Traffic/CBR]
$cbr_(14) set packetSize_ 512
$cbr_(14) set interval_ 0.25
$cbr_(14) set random_ 1
```

77

```
$cbr_(14) set maxpkts_ 10000
$cbr_(14) attach-agent $udp_(14)
$ns_ connect $udp_(14) $null_(14)
$ns_ at 8.5436124673781979 "$cbr_(14) start"
#
# 19 connecting to 20 at time 59.082160703410466
set udp_(15) [new Agent/UDP]
$ns_ attach-agent $node_(19) $udp_(15)
set null_(15) [new Agent/Null]
$ns_ attach-agent $node_(20) $null_(15)
set cbr_(15) [new Application/Traffic/CBR]
$cbr_(15) set packetSize_ 512
$cbr_(15) set interval_ 0.25
$cbr_(15) set random_ 1
$cbr_(15) set maxpkts_ 10000
$cbr_(15) attach-agent $udp_(15)
$ns_ connect $udp_(15) $null_(15)
$ns_ at 59.082160703410466 "$cbr_(15) start"
#
# 20 connecting to 21 at time 136.15388747125581
set udp_(16) [new Agent/UDP]
$ns_ attach-agent $node_(20) $udp_(16)
set null_(16) [new Agent/Null]
$ns_ attach-agent $node_(21) $null_(16)
set cbr_(16) [new Application/Traffic/CBR]
$cbr_(16) set packetSize_ 512
$cbr_(16) set interval_ 0.25
$cbr_(16) set random_ 1
$cbr_(16) set maxpkts_ 10000
$cbr_(16) attach-agent $udp_(16)
$ns_ connect $udp_(16) $null_(16)
$ns_ at 136.15388747125581 "$cbr_(16) start"
#
# 21 connecting to 22 at time 65.760960730612723
set udp_(17) [new Agent/UDP]
$ns_ attach-agent $node_(21) $udp_(17)
set null_(17) [new Agent/Null]
$ns_ attach-agent $node_(22) $null_(17)
set cbr_(17) [new Application/Traffic/CBR]
$cbr_(17) set packetSize_ 512
$cbr_(17) set interval_ 0.25
$cbr_(17) set random_ 1
$cbr_(17) set maxpkts_ 10000
$cbr_(17) attach-agent $udp_(17)
```

```
$ns_ connect $udp_(17) $null_(17)
$ns_ at 65.760960730612723 "$cbr_(17) start"
#
# 21 connecting to 23 at time 44.466999408075118
set udp_(18) [new Agent/UDP]
$ns_ attach-agent $node_(21) $udp_(18)
set null_(18) [new Agent/Null]
$ns_ attach-agent $node_(23) $null_(18)
set cbr_(18) [new Application/Traffic/CBR]
$cbr_(18) set packetSize_ 512
$cbr_(18) set interval_ 0.25
$cbr_(18) set random_ 1
$cbr_(18) set maxpkts_ 10000
$cbr_(18) attach-agent $udp_(18)
$ns_ connect $udp_(18) $null_(18)
$ns_ at 44.466999408075118 "$cbr_(18) start"
#
# 22 connecting to 23 at time 130.07887213960237
set udp_(19) [new Agent/UDP]
$ns_ attach-agent $node_(22) $udp_(19)
set null_(19) [new Agent/Null]
$ns_ attach-agent $node_(23) $null_(19)
set cbr_(19) [new Application/Traffic/CBR]
$cbr_(19) set packetSize_ 512
$cbr_(19) set interval_ 0.25
$cbr_(19) set random_ 1
$cbr_(19) set maxpkts_ 10000
$cbr_(19) attach-agent $udp_(19)
$ns_ connect $udp_(19) $null_(19)
$ns_ at 130.07887213960237 "$cbr_(19) start"
#
#Total sources/connections: 14/20
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

1. Operational Requirements Document (ORD) for Joint Tactical Radio System (JTRS), JTRS Joint Program Office, 23 March 1998.

2. Corson, Scott S., Macker, J., "Mobile Ad Hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations," RFC 2501, January 1999.

3. Gerla, M., Lee, S. J., Toh, C. K., "A Simulation Study of Table-Driven and On-Demand Routing Protocols for Mobile Ad Hoc Networks," *IEEE Network*, Vol 13 Issue 4, pg 48-54, Jul-Aug 1999.

4. Maatta, Risto, "Wireless Ad Hoc Routing Protocols, a Taxonomy," Defense Forces Research Institute of Technology, Electronics and Information Technology Seminar, pg 1-19, 11 May 2000.

5. Misra, Padmini, "Routing Protocols For Ad Hoc Mobile Wireless Networks," Computer and Information Systems Paper 788-99, Ohio State University, 18 November 1999.

6. Bhagwat, P. and Perkins, Charles E., "Highly Dyamic Destination Sequenced Distance Vector (DSDV) Routing for Mobile Computers," *Proceedings of the SIGCOMM Conference on Communications Architectures, Protocols and Applications*, pg 234-244, August 1994.

7. Lesiuk, Camberon B., "Routing in Ad Hoc Networks of Mobile Hosts," Directed Study, Department of Mechanical Engineering, University of Victoria, Victoria BC, Canada, 2 December 1998.

8. Vaidya, Nitin H., "Mobile Ad Hoc Networks; Routing, MAC, and Transport Issues," MobiComm Tutorial, 15 July, 2000, pg 1- 431.

9. Royer, Elizabeth M., Toh Chai-Keong, "A Review of Current Routing Protocols for Ad Hoc Mobile Wireless Networks," *IEEE Personal Communications*, April 1999.

10. Broch J., Hu, Y. C., Jetcheva J., Johnson D. B., Maltz, D. A., "A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols," *Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 1998)*, Dallas, TX, October 1998.

11. Castaneda, R., Das, Samir R., Yan, J. "Simulation-Based Performance Evaluation of Routing Protocols for Mobile Ad Hoc Networks," *Mobile Networks and Applications*, Vol 5, Issue 3, pg 179-189. July 2000.

12. Broch J., Johnson D. B., Maltz, D. A., "The Dyanmic Source Routing (DSR) Protocol for Mobile Ad Hoc Networks," Internet Draft, draft-ietf-manet-dsr-01.txt, Dec 1998.

13. Das, Samir R., Perkins, Charles E., Royer, Elizabeth M., "Performance Comparison of Two On-Demand Routing Protocols for Ad Hoc Networks," *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, Tel Aviv, Israel, March 2000, pg. 3-12.

14. Das, Samir R., Perkins, Charles E., Royer, Elizabeth M., "The Ad-hoc On-Demand Distance Vector Routing Protocol (AODV) for Ad Hoc Networks," Internet Draft, draft-ietf-manet-aodv-06.txt, July 2000.

15. Perkins, Charles E., Royer, Elizabeth M., "Ad-hoc On-Demand Distance Vector Routing," *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, New Orleans, LA, February 1999, pp. 90-100.

16. Johnson, David B., "Routing in Ad Hoc Networks," *Proceedings of IEEE Workshop*, pg 1-4, 1994.

17. Network Simulator Notes and Documentation, The VINT Project, August 2000.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center 2
   8725 John J. Kingman Road, Ste 0944
   Fort Belvoir, VA 22060-6218

2. Dudley Knox Library... 2
   Naval Postgraduate School
   411 Dyer Road
   Monterey, California 93943-5101

3. Director, Training and Education... 1
   MCCDC, Code C46
   1019 Elliot Rd.
   Quantico, Virginia 22134-5027

4. Director, Marine Corps Research Center 2
   MCCDC, Code C40RC
   2040 Broadway Street
   Quantico, Virginia 22134-5107

5. Marine Corps Tactical System Support Activity..................................... 4
   Technical Advisory Branch
   Attn: Librarian
   Box 555171
   Camp Pendleton, CA 92055-5080

6. Marine Corps Representative 1
   Naval Postgraduate School
   Code 037, Bldg. 330, Ingersoll Hall, Room 116
   555 Dyer Road
   Monterey, CA 93943

7. Chairman, Code EC... 1
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, California 93943-5121

8. Professor Murali Tummala, Code EC/Tu 2
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, California 93943-5121